

Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia Eletrônica

**Implementação em SoC FPGA dos Módulos de
Codificação e MF-TDMA do Protocolo
DVB-RCS2**

Autor: Bruno Alves Ferreira Camargos
Orientador: Prof. Dr. Daniel Mauricio Muñoz Arboleda

Brasília, DF
2022



Bruno Alves Ferreira Camargos

Implementação em SoC FPGA dos Módulos de Codificação e MF-TDMA do Protocolo DVB-RCS2

Monografia submetida ao curso de graduação em (Engenharia Eletrônica) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia Eletrônica).

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. Dr. Daniel Mauricio Muñoz Arboleda

Brasília, DF

2022

Bruno Alves Ferreira Camargos

Implementação em SoC FPGA dos Módulos de Codificação e MF-TDMA do
Protocolo DVB-RCS2/ Bruno Alves Ferreira Camargos. – Brasília, DF, 2022-
107 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Daniel Mauricio Muñoz Arboleda

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2022.

1. Hardware Reconfigurável. 2. . I. Prof. Dr. Daniel Mauricio Muñoz Arboleda. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Implementação em SoC FPGA dos Módulos de Codificação e MF-TDMA do Protocolo DVB-RCS2

CDU 02:141:005.6

Bruno Alves Ferreira Camargos

Implementação em SoC FPGA dos Módulos de Codificação e MF-TDMA do Protocolo DVB-RCS2

Monografia submetida ao curso de graduação em (Engenharia Eletrônica) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia Eletrônica).

Trabalho aprovado. Brasília, DF, 10 de maio de 2022:

**Prof. Dr. Daniel Mauricio Muñoz
Arboleda**
Orientador

Prof. Dr. Daniel Araujo
Convidado 1

Prof. Dr. Gilmar Silva Beserra
Convidado 2

Brasília, DF
2022

Dedico este trabalho de fim de curso à...

Agradecimentos

Aos professores, pelo conhecimento transmitido e pelos desafios propostos.

Ao meu orientador, pelo apoio e confiança.

Aos amigos e colegas, pelo apoio e pelos conselhos.

A todos que contribuíram direta ou indiretamente para a minha formação

*“Meu nome é Ozymandias, rei dos reis:
Contemplai as minhas obras, ó poderosos e desesperai-vos!
(Percy Bysshe Shelley, Ozymandias 12, 2)*

Resumo

Este trabalho apresentará estudos e implementações de módulos do protocolo *Digital Video Broadcasting - Return Channel via Satellite* (DVB-RCS2). O protocolo DVB-RCS2 é utilizado principalmente para transmissão satelital de TV digital, sendo capaz de prover conexão a usuários em tempo real em diversos tipos, como texto, voz e imagens. Desta forma, este sistema é capazes de realizar uma comunicação com uma central chamada HUB, e a partir das informações obtidas desta, o sistema deve ser capaz de codificar um sinal e realizar a sua transmissão. O protocolo DVB-RCS2 (ETSI, 2014-04) apresenta todas as etapas que envolvem a construção do sistema. Entretanto, as que serão abordadas neste trabalho são: *modulation*, *spreading* e o *Multi-frequency time-division multiple access*. Cada um destes módulos possui uma função específica no sistema. Os blocos de modulação são necessários para a correta codificação dos dados a serem transmitidos, tendo como base as diferentes taxas de códigos e os 4 tipos de modulação BPSK, QPSK, 8PSK e 16-QAM. O módulo de *spreading* é necessário para uma codificação mais robusta do sinal, tendo em vista sistemas em que a relação sinal ruído é muito baixa. O MF-TDMA é o módulo de controle do sistema, o qual realiza a comunicação com a HUB e recebe informações necessárias ao funcionamento do sistema em forma de tabelas e descritores e os utiliza para controle de transmissão e codificação do sistema. O trabalho apresenta a implementação do MF-TDMA aderente ao protocolo DVB-RCS2 em ARM, e a sua verificação foi feita a partir da constatação do correto funcionamento das transições de estado do sistema e do funcionamento de suas funções internas. A aplicação em Labview e o python serão utilizados para a visualização do correto funcionamento da maquina de estados do sistema, bem como das funções de aquisição de *timeslots* para transmissão. A implementação dos módulos *spreading* e *modulation* do codificador RCS2 foi feita em ARM e foi validada a partir de modelos implementados em python e octave. Como referencia para a implementação do bloco *modulation*, serão utilizado trabalhos anteriores que implementaram o filtro *Square-Root Raised Cosine* (SRRC). O bloco de sincronismo de relógio do MF-TDMA foi implementado tendo como referencia 2 trabalhos que implementaram o sistema a partir de um sistema *Phase Locked Loop* (PLL). Os resultados alcançados incluem quanto ao codificador RCS2 a comparação do sistema implementado em ARM com os modelos de referencia em Python, o tempo de execução de cada um dos módulos desenvolvidos e a demonstração prática usando rádios definidos por software (SDRs) do correto funcionamento dos estados que foram codificados do MF-TDMA.

Palavras-chaves: DVB-RCS2. MF-TDMA. NCR. FPGA. Rádio Definido por Software.

Abstract

This work will present studies and implementations of modules of the *Digital Video Broadcasting - Return Channel via Satellite* (DVB-RCS2) protocol. The DVB-RCS2 protocol is mainly used for satellite transmission of digital TV, being able to provide real-time connection to users in different types, such as text, voice and images. In this way, this system is able to communicate with a central called HUB, and from the information obtained from it, the system must be able to encode a signal and carry out its transmission. The DVB-RCS2 (ETSI, 2014-04) protocol presents all the steps involved in building the system. However, the ones that will be addressed in this work are: *modulation*, *spreading* and *Multi-frequency time-division multiple access*. Each of these modules has a specific function in the system. The modulation blocks are necessary for the correct encoding of the data to be transmitted, based on the different code rates and the 4 types of modulation BPSK, QPSK, 8PSK and 16-QAM. The *spreading* module is necessary for a more robust encoding of the signal, considering systems where the signal to noise ratio is very low. The MF-TDMA is the system's control module, which communicates with the HUB and receives information necessary for the system to function in the form of tables and descriptors and uses them for transmission control and system coding. The work presents the implementation of MF-TDMA adhering to the DVB-RCS2 protocol in ARM, and its verification will be made from the verification of the correct functioning of the system state transitions and the functioning of its internal functions. Labview software and python will be used to visualize the correct functioning of the system state machine, as well as the *timeslots* acquisition functions for transmission. The implementation of the modules *spreading* and *modulation* of the RCS2 encoder will be done in ARM and will be validated from models implemented in python and octave, where the correct coding of each one of the modules will be verified. As a reference for the implementation of the *modulation* block, previous works that implemented the *Square-Root Raised Cosine* (SRRC) filter will be used. The MF-TDMA clock synchronization block will be implemented with reference to 2 works that implemented the system from a *Phase Locked Loop* (PLL) system. The results achieved include, regarding the RCS2 encoder, the comparison of the system implemented in ARM with the reference models in Python, the execution time of each of the developed modules and the demonstration of the correct functioning of the states that were encoded in the MF-TDMA.

Key-words: DVB-S2x. DVB-S2. Reconfigurable Architecture. Frequency Synchronization. Phase Synchronization. FPGA. VHDL.

Lista de Figuras

| | |
|---|----|
| Figura 1 – Arquitetura básica do Terminal DVB-RCS2. Retirado de (ERL; COLA, 2014) | 25 |
| Figura 2 – Blocos do Codificador DVB-RCS2. Retirado de (ERL; COLA, 2014) | 25 |
| Figura 3 – Resposta ao impulso do filtro SRRC retirado de (JOOST, 2010) | 27 |
| Figura 4 – Mapeamento de bit para símbolo: Modulação $\pi/2$ -BPSK.(ETSI, 2014-04) | 28 |
| Figura 5 – Constelação: Modulação $\pi/2$ -BPSK(ETSI, 2014-04) | 28 |
| Figura 6 – Mapeamento de bit para símbolo: Modulação QPSK.(ETSI, 2014-04) | 29 |
| Figura 7 – Constelação: Modulação QPSK.(ETSI, 2014-04) | 29 |
| Figura 8 – Mapeamento de bit para símbolo: Modulação 8PSK - rate 2/3.(ETSI, 2014-04) | 30 |
| Figura 9 – Mapeamento de bit para símbolo: Modulação 8PSK - rate 5/6 e 3/4.(ETSI, 2014-04) | 30 |
| Figura 10 – Constelação: Modulação 8-BPSK.(ETSI, 2014-04) | 30 |
| Figura 11 – Mapeamento de bit para símbolo: Modulação 16-QAM.(ETSI, 2014-04) | 31 |
| Figura 12 – Mapeamento da constelação: Modulação 16-QAM(ETSI, 2014-04) | 31 |
| Figura 13 – Mapeamento da constelação: Modulação 16-QAM (NETTO, 2009). | 32 |
| Figura 14 – Sinal em banda estreita e com espalhamento (CS457/CS546. . . , 2001). | 33 |
| Figura 15 – Exemplo para geração de sequência de embaralhamento (ETSI, 2014-04). | 33 |
| Figura 16 – Super Frame - Frames disponíveis (ETSI, 2014-04) | 34 |
| Figura 17 – Frame: Slots disponíveis(ETSI, 2014-04) | 35 |
| Figura 18 – Timeslots e BTUs(ETSI, 2014-04) | 35 |
| Figura 19 – Diagrama de um sistema PLL | 36 |
| Figura 20 – Arquitetura geral de uma FPGA Zynq Ultrascale (ABUOWAIMER DANI MAAROUF; VANNELLI, 2018) | 37 |
| Figura 21 – SoC FPGA Zynq-7000 (XILINX,) | 38 |
| Figura 22 – Arquitetura do filtro RRC (SATEESHKUMAR, 2013) | 39 |
| Figura 23 – Magnitude do filtro RRC (SATEESHKUMAR, 2013) | 40 |
| Figura 24 – Arquitetura geral I do MF-TDMA proposta em (JEONG JOONGYU RYU, 2003) | 41 |
| Figura 25 – Arquitetura geral II do MFTDMA proposta em (JEONG JOONGYU RYU, 2003) | 41 |
| Figura 26 – Diagrama de blocos do DSSS proposta em (R.PRABHU R.NAGARAJAN, 2017) | 42 |
| Figura 27 – Diagrama de blocos do CD-MA sugerida em (SAROJINI, 2012) | 43 |
| Figura 28 – Diagrama de blocos do CDMA sugerida em (SAROJINI, 2012) | 43 |
| Figura 29 – Diagrama de blocos proposto em(LISS, 2012) | 44 |

| | |
|--|----|
| Figura 30 – Medições no tempo dos valores recebidos e corrigidos de PCR(LISS, 2012) | 45 |
| Figura 31 – Diagrama de blocos do sistema PLL para recuperação de relógio proposto em (JEON, 2014) | 45 |
| Figura 32 – Diagrama de blocos do módulo <i>Modulation</i> do codificador RCS2 | 47 |
| Figura 33 – Diagrama de blocos do módulo <i>spreading</i> do codificador RCS2 | 48 |
| Figura 34 – Inserção de símbolos de preâmbulo e pós-âmbulo do bloco <i>INS.SYMBBS</i> do codificador RCS2. Retirado de (ETSI, 2014-04) | 48 |
| Figura 35 – sequencia padrão de espalhamento do codificador RCS2 retirado de (ETSI, 2014-04) | 49 |
| Figura 36 – Principais tabelas utilizadas no MF-TDMA (ETSI, 2014-04) | 49 |
| Figura 37 – MF-TDMA: FSM principal (ETSI, 2014-04) | 51 |
| Figura 38 – Ready for Logon(ETSI, 2014-04) | 54 |
| Figura 39 – Acesso randômico | 55 |
| Figura 40 – Ready for TDMA sync(ETSI, 2014-04) | 56 |
| Figura 41 – TDMA Sync(ETSI, 2014-04) | 56 |
| Figura 42 – Arquitetura geral do sistema (ETSI, 2014-04) | 57 |
| Figura 43 – Algoritmo de recepção dos ts packet | 58 |
| Figura 44 – filtro passa baixas | 58 |
| Figura 45 – Kit de desenvolvimento Zybo Zynq-7000 ARM/FPGA SoC Trainer Board (DIGILENT,) | 60 |
| Figura 46 – Simulação de uma HUB utilizando o GNU radio. | 63 |
| Figura 47 – Bloco de controle das transmissões do MF-TDMA | 65 |
| Figura 48 – Arquitetura para a transmissão no tempo | 66 |
| Figura 49 – TIME SYNC - FSM | 67 |
| Figura 50 – TIME SYNC IP | 67 |
| Figura 51 – Interface RTL do bloco de sincronia | 68 |
| Figura 52 – Interface RTL do contador local | 69 |
| Figura 53 – Interface RTL do detector de fase | 69 |
| Figura 54 – Interface RTL do filtro IRR | 70 |
| Figura 55 – Interface RTL do contador local | 70 |
| Figura 56 – Interface RTL do contador local | 71 |
| Figura 57 – Arquitetura de teste do sincronizador | 71 |
| Figura 58 – octave: $\pi/2$ bpsk | 72 |
| Figura 59 – ARM: $\pi/2BPSK$ | 72 |
| Figura 60 – octave: QPSK | 72 |
| Figura 61 – ARM: QPSK | 72 |
| Figura 62 – octave: 8PSK | 73 |
| Figura 63 – ARM: 16QAM | 73 |

| | |
|---|----|
| Figura 64 – octave: 16QAM. | 73 |
| Figura 65 – ARM: 16QAM | 73 |
| Figura 66 – Diagrama de olho: pi/2 BPSK | 74 |
| Figura 67 – Diagrama de olho: QPSK | 74 |
| Figura 68 – Diagrama de olho: 8PSK | 74 |
| Figura 69 – Diagrama de olho: 16QAM | 74 |
| Figura 70 – Constelação: pi/2 BPSK | 75 |
| Figura 71 – Constelação: QPSK | 75 |
| Figura 72 – Constelação: 8PSK | 75 |
| Figura 73 – Constelação: 16QAM | 75 |
| Figura 74 – Transmissão: Tx Begin | 76 |
| Figura 75 – Transmissão: Tx End | 76 |
| Figura 76 – Sincronizador: Sem sincronia A | 77 |
| Figura 77 – Sincronizador: Sincronia recuperada A | 77 |
| Figura 78 – Sincronizador: Sem sincronia B | 78 |
| Figura 79 – Sincronizador: Sincronia recuperada B | 78 |
| Figura 80 – Formas de ondas sobrepostas da implementação em ARM com modelo em python - BPSK | 79 |
| Figura 81 – Formas de ondas sobrepostas da implementação em ARM com modelo em python - QPSK | 80 |
| Figura 82 – Formas de ondas sobrepostas da implementação em ARM com modelo em python - 8PSK | 80 |
| Figura 83 – Tempo de execução: <i>Linear Modulation</i> | 83 |
| Figura 84 – Tempo de execução: <i>Spread Spectrum Linear Modulation</i> | 84 |
| Figura 85 – MF-TDMA: NCR Recovery | 85 |
| Figura 86 – Sincronizador: Perda de sincronia após 5 segundos | 85 |
| Figura 87 – Sincronizador: Perda de sincronia após 10 segundos | 86 |
| Figura 88 – Sincronizador: Consumo de recursos | 87 |
| Figura 89 – MF-TDMA: Off/standby | 88 |
| Figura 90 – MF-TDMA: Ready for logon | 88 |
| Figura 91 – MF-TDMA: Ready for tdma sync | 89 |
| Figura 92 – MF-TDMA: tdma sync | 89 |
| Figura 93 – Mf-TDMA: Setup experimental | 91 |
| Figura 94 – MF-TDMA: Hold Standby | 92 |
| Figura 95 – MF-TDMA: Off Standby | 92 |
| Figura 96 – MF-TDMA: Ready for logon - Timeslot 1 | 93 |
| Figura 97 – GNU Radio: Frequencia de RX em 2.2 Ghz - Timeslot 1 | 93 |
| Figura 98 – MF-TDMA: Ready for TDMA - Timeslot 2 | 94 |
| Figura 99 – GNU Radio: Frequencia de RX em 2.2 Ghz - Timeslot 2 | 94 |

| | |
|--|-----|
| Figura 100–MF-TDMA: TDMA SYNC - Timeslot 3 | 95 |
| Figura 101–GNU Radio: Frequencia de RX em 2.2 Ghz - Timeslot 3 | 95 |
| Figura 102–MF-TDMA:TDMA SYNC - Timeslot 4 | 96 |
| Figura 103–GNU Radio: Frequencia de RX em 2.2 Ghz - Timeslot 4 | 96 |
| Figura 104–MF-TDMA: Ready for TDMA - Timeslot 5 | 97 |
| Figura 105–GNU Radio: Frequencia de RX em 2.2 Ghz - Timeslot 5 | 97 |
| Figura 106–MF-TDMA: TDMA SYNC - Timeslot 6 | 98 |
| Figura 107–GNU Radio: Frequencia de RX em 2.325 Ghz - Timeslot 6 | 98 |
| Figura 108–MF-TDMA: TDMA SYNC - Timeslot 7 | 99 |
| Figura 109–GNU Radio: Frequencia de RX em 2.325 Ghz - Timeslot 7 | 99 |
| Figura 110–MF-TDMA: TDMA SYNC - Timeslot 8 | 100 |
| Figura 111–GNU Radio: Frequencia de RX em 2.325 Ghz - Timeslot 8 | 100 |
| Figura 112–MF-TDMA: TDMA SYNC - Timeslot 9 | 101 |
| Figura 113–GNU Radio: Frequencia de RX em 2.325 Ghz - Timeslot 9 | 101 |
| Figura 114–MF-TDMA: Validação do tempo pelo LED | 103 |
| Figura 115–MF-TDMA: Validação do tempo pelo LED | 103 |

Lista de Tabelas

| | |
|---|----|
| Tabela 1 – Detalhes da implementação de blocos do protocolo DVB-RCS2: Modulação, spread-spectrum e MF-TDMA. | 46 |
| Tabela 2 – Condições de teste do MF-TDMA | 62 |
| Tabela 3 – Super frame de teste | 62 |
| Tabela 4 – Condições de teste do MF-TDMA no GNU radio | 63 |
| Tabela 5 – Reconfiguração dos <i>timeslot</i> | 64 |
| Tabela 6 – Super Frame | 64 |
| Tabela 7 – Frame | 64 |

Lista de abreviaturas e siglas

| | |
|---------|--|
| 8-PSK | <i>Eight Phase Shift Keying</i> |
| 16-QAM | <i>Sixteen Quadrature Amplitude Modulation</i> |
| ADC | <i>Analog-Digital Converter</i> |
| BCT | <i>RL Broadcast Configuration Table</i> |
| BPSK | <i>Binary Phase Shift Keying</i> |
| BPSK | <i>Phase-shift keying</i> |
| BTU | <i>Bandwidth-Time Unit</i> |
| CMT | <i>Correction Message Table</i> |
| DA | <i>Dedicated Access</i> |
| DDS | <i>Direct Digital Synthesis</i> |
| DM | <i>Delay-Multiply</i> |
| DVB | <i>Digital Video Broadcasting</i> |
| FCT2 | <i>Frame Configuration Table 2</i> |
| FL | <i>Forward Link</i> |
| FFT | <i>Fast Fourier Transform</i> |
| FL | <i>Forward Link</i> |
| FSM | <i>Finite State Machine</i> |
| HID | <i>Hardware Identifier</i> |
| HW | <i>HardWare</i> |
| ISI | <i>Inter Symbol Interference</i> |
| LM | <i>Linear Modulation (or Modulator)</i> |
| MF-TDMA | <i>Multi-Frequency TDMA</i> |
| MPEG | <i>Moving Pictures Expert Group</i> |

| | |
|---------|---|
| MSPS | <i>Mega Símbolo Por Segundo</i> |
| NCC | <i>Network Control Centre</i> |
| NCR | <i>Network Clock Reference</i> |
| PC | <i>Phase Correction</i> |
| QPSK | <i>Quadrature Phase-Shift Keying</i> |
| RCS | <i>Return Channel over Satellite</i> |
| RL | <i>Return Link</i> |
| RMSE | <i>Root-Mean-Square Error</i> |
| RMS | <i>Root-Mean-Square</i> |
| RRC | <i>Root-Rised-Cosine</i> |
| SCT | <i>Superframe Composition Table</i> |
| SoC | <i>System on Chip</i> |
| SOF | <i>Start Of Frame</i> |
| TC | <i>Turbo Coding</i> |
| TC-SSLM | <i>Turbo Coding Spread Spectrum Linear Modulation</i> |
| TC-LM | <i>Turbo Coding Linear Modulation</i> |
| TDMA | <i>Time Division Multiple Access</i> |
| TDM | <i>Time Division Multiplex</i> |
| TIM-B | <i>Terminal Information Message Broadcas</i> |
| TIM-U | <i>Terminal Information Message Unicast</i> |
| VHDL | <i>VHSIC Hardware Description Language</i> |
| VHSIC | <i>Very High Speed Integrated Circuits</i> |
| | <i>BPSK Binary PSK</i> |
| | <i>BTU Bandwidth-Time Unit</i> |
| | <i>BW Bandwidth</i> |
| | <i>NCC Network Control Centre</i> |

NCR Network Clock Reference

NIT Network Information Table

QAM Quadrature Amplitude Modulation

RCST RCS Terminal

Lista de símbolos

| | |
|-----------|---|
| dB | Decibel |
| E_s/N_0 | Razão da energia do símbolo pela energia do ruído |
| SnR | Relação Sinal-Ruído |
| θ | Theta |
| σ | Sigma |
| ζ | Zeta |

Sumário

| | | |
|----------|--|-----------|
| 1 | INTRODUÇÃO | 21 |
| 1.1 | Contextualização | 21 |
| 1.2 | Justificativa | 22 |
| 1.3 | Objetivos | 22 |
| 1.4 | Objetivos Específicos | 22 |
| 1.5 | Organização do documento | 23 |
| 2 | FUNDAMENTAÇÃO TEÓRICA | 24 |
| 2.1 | Protocolo DVB-RCS2 | 24 |
| 2.2 | Moduladores | 25 |
| 2.2.1 | Sobre-amostrador | 26 |
| 2.2.2 | Filtro raiz quadrada do cosseno levantado | 26 |
| 2.2.3 | Esquemas de modulação | 27 |
| 2.2.4 | $\pi/2$ -BPSK | 27 |
| 2.2.5 | QPSK | 29 |
| 2.2.6 | 8PSK | 30 |
| 2.2.7 | 16QAM | 31 |
| 2.3 | Espalhamento de espectro | 32 |
| 2.3.1 | Banda estreita | 32 |
| 2.3.2 | Gerador de sequências | 32 |
| 2.3.3 | Espalhamento das frequências | 32 |
| 2.3.4 | Sistema utilizado no protocolo DVB-RCS2 | 33 |
| 2.4 | MF-TDMA | 34 |
| 2.4.1 | Super Frame | 34 |
| 2.4.2 | Frame | 35 |
| 2.4.3 | Timeslot | 35 |
| 2.4.4 | Tempo de guarda | 35 |
| 2.4.5 | Sincronização de relógio | 36 |
| 2.4.6 | Malha de captura de fase | 36 |
| 2.5 | Hardware Reconfigurável e Sistemas em Chip (SoC) | 37 |
| 2.6 | Estado da Arte | 39 |
| 2.6.1 | Filtro SRRC | 39 |
| 2.6.2 | MF-TDMA | 40 |
| 2.6.3 | CDMA | 42 |
| 2.6.4 | Recuperação de sincronismo | 44 |

| | | |
|------------|--|-----------|
| 3 | METODOLOGIA E FERRAMENTAS | 47 |
| 3.1 | <i>Codificador RCS2</i> | 47 |
| 3.1.1 | <i>Esquemas de modulação</i> | 47 |
| 3.1.2 | <i>Spreading</i> | 48 |
| 3.2 | <i>Multi-frequency time-division multiple access</i> | 49 |
| 3.2.1 | Tabelas e descritores | 49 |
| 3.2.2 | Máquina de estado principal | 50 |
| 3.2.3 | Algoritmo de <i>logon</i> | 53 |
| 3.2.4 | Algoritmo de sincronização | 55 |
| 3.2.5 | Sincronismo de relógio - NCR | 56 |
| 3.2.5.1 | Recepção dos TS-packet | 56 |
| 3.2.5.2 | Controlador | 57 |
| 3.2.5.3 | System Time Clock (STC) | 59 |
| 3.2.5.4 | Contador Local | 59 |
| 3.3 | Ferramentas | 59 |
| 4 | PROPOSTA DE VALIDAÇÃO E IMPLEMENTAÇÃO | 61 |
| 4.1 | <i>Modulation and Spreading</i> | 61 |
| 4.2 | <i>Multi-frequency time-division multiple access</i> | 61 |
| 4.2.1 | Controle do tempo e validação | 64 |
| 4.2.2 | Verificação na frequência de transmissão | 66 |
| 4.3 | <i>Sincronismo de relógio - NCR</i> | 66 |
| 4.3.1 | <i>ADPLL</i> | 68 |
| 4.3.2 | <i>Contador Local</i> | 69 |
| 4.3.3 | <i>Detector de fase</i> | 69 |
| 4.3.4 | <i>Filtro IRR</i> | 69 |
| 4.3.5 | <i>Palavra de Frequência</i> | 70 |
| 4.3.6 | <i>DDS</i> | 70 |
| 4.3.7 | <i>Arquitetura de teste</i> | 71 |
| 5 | RESULTADOS | 72 |
| 5.1 | simulação dos moduladores | 72 |
| 5.1.1 | Sinais em fase e quadratura filtrados | 72 |
| 5.1.2 | Diagrama de olho | 73 |
| 5.1.3 | Constelação | 74 |
| 5.2 | Simulação comportamental do bloco de transmissão | 76 |
| 5.3 | Simulação do bloco de sincronização | 77 |
| 5.4 | Verificação da integração RCS2: <i>Modulador, Spreading, filtro</i> | 79 |
| 5.5 | Sincronizador | 84 |
| 5.5.1 | Consumo de recursos de hardware do módulo sincronizador | 86 |

| | | |
|------------|--|------------|
| 5.6 | <i>Verificação do Multi-frequency time-division multiple access</i> | 87 |
| 5.6.1 | Resultados sem emulação de HUB | 87 |
| 5.6.2 | Resultados com emulação de HUB | 90 |
| 6 | CONCLUSÃO | 104 |
| | REFERÊNCIAS | 106 |

1 Introdução

1.1 Contextualização

O padrão europeu para TV digital DVB, fornece uma cadeia robusta e de alta velocidade, a qual consegue fornecer vários *megabits* por segundo para diversos canais de TV digital MPEG-2.

Com as tecnologias antigas, os *transponders* de um satélite suportavam apenas um canal de transmissão. A tecnologia de TV digital DVB, permitiu que a taxa de transmissão fosse aumentada, de forma que um *transponder* consiga transmitir em torno de 40 Megabits por segundo. Ademais, com as tecnologias de multiplexação, um mesmo *transponder* pode transmitir 10 canais, de forma que, supondo uma taxa de (40Mb/s), cada canal tenha entorno de 4MB/s(ETSI, 2014-04).

Neste projeto foram adotadas as normas para transmissão satelital de TV digital DVB, as quais apresentam o processo de codificação e transmissão de um sinal digital e o processo de desenvolvimento do protocolo de comunicação via satélite com uma central de comando. O protocolo DVB-RCS2 consiste em um padrão de comunicação *open source* altamente eficiente no gerenciamento de banda de um sistema utilizado em transmissão satelital de TV digital. No DVB-RCS2, cada tipo de transmissão é realizada com uma finalidade específica, seja para transmissão de dados relativos à posição, vídeos, mensagens de texto ou voz, dados de sensores, enter outros. Cada uma dessas categorias de transmissão podem ser feitos adotando uma codificação específica, a qual aumenta a qualidade e confiabilidade da transmissão.

Durante a transmissão de um sinal, diversos fatores influenciam o sinal, tais como desvios de frequência da portadora, perda de sincronismo entre fonte e receptor. Ademais, as condições ambientais afetam o sinal transmitido, de forma a alterar a relação sinal-ruído do canal. Desta forma, um sistema capaz de identificar a codificação correta para cada meio é necessário para garantir maior confiabilidade durante uma transmissão.

Considerando que vários sistemas se comuniquem em tempo real com uma central, diversos problemas relacionados a sincronização, identificação e recepção de sinais podem surgir, pois, a central pode não conseguir lidar com um alto fluxo de dados. Desta forma, uma solução que melhore o desempenho e mantenha o sincronismo entre os terminais e a HUB deve ser implementada de modo a garantir a robustez do sistema de comunicação satelital.

1.2 Justificativa

Neste trabalho, um sistema de controle MF-TDMA, foi implementado fazendo o uso de um co-projeto hardware software. O sistema foi construído para lidar com tomadas de decisões por parte de uma máquina de estados finita (FSM) a partir consulta de tabelas e descritores cujas informações são enviadas pela HUB, conforme explicado na norma (ETSI, 2014-04).

A FSM principal e os algoritmos internos são construídos para lidar realizar tomadas de decisões que envolvem a leitura e escrita em blocos de memória, as quais são executadas de forma rápida pelo processador ARM. O bloco crítico do sistema foi implementado em hardware usando linguagem VHDL. Este bloco realiza a recuperação de sincronismo de relógio do MF-TDMA. A implementação do bloco de codificação do sistema RCS2 deve cumprir com o critério de tempo de execução máximo $40ms$ para uma modulação linear.

Para o desenvolvimento adotou-se a utilização de dispositivos FPGAs (*Field Programmable Gate Arrays*). A seleção de FPGAs se justifica pela capacidade de alcance de altas taxas de transmissão e processamento alcançadas por esses dispositivos, além da flexibilidade dos FPGAs em razão da possibilidade de reconfiguração do textithardware. Os módulos do protocolo DVB-RCS2 implementados no presente trabalho estão relacionados com o processo de transmissão, especificamente os blocos *Modulation* e *Spreading* e o módulo de controle, ou MF-TDMA.

1.3 Objetivos

O objetivo deste trabalho consiste na implementação em ARM dos módulos *spreading* e *modulator* do codificador DVB-RCS2, bem como a implementação do MF-TDMA usando a técnica de co-projeto textithardware software em FPGA.

1.4 Objetivos Específicos

- Implementação em ARM do MF-TDMA aderente ao protocolo DVB-RCS2.
- Implementação do codificador DVB-RCS2 em ARM.
- Integração do codificador DVB-RCS2 com o Transceptor AD9361.
- Implementação do bloco de sincronia de relógio (NCR) em textithardware.
- Integrar o MF-TDMA com o NCR através de um barramento AXI4-LITE.
- Integrar o MF-TDMA com o codificador em ARM.

1.5 Organização do documento

O restante deste documento está organizado da seguinte maneira: O capítulo 2 versa sobre a fundamentação teórica, onde serão apresentados conceitos do protocolo DVB-RCS2, do MF-TDMA, moduladores, *spreading* e também do estado da arte, que apresenta trabalhos de implementações semelhantes destes módulos. O capítulo 3 trata sobre toda a metodologia e ferramentas propostas na implementação dos módulos do MF-TDMA, moduladores e *spreading* e como os blocos que serão implementados. O capítulo 4 trata sobre a forma que foi implementado os módulos apresentados no capítulo 3 e a forma de validação, definições de arquitetura. Já no capítulo 4 serão apresentados os resultados obtidos pela simulação e validação dos módulos aqui implementados. Por fim, têm-se as conclusões do trabalho, onde serão discutidos os resultados.

2 Fundamentação Teórica

2.1 Protocolo DVB-RCS2

O protocolo DVB-RCS2 consiste em um padrão para uma conexão de banda larga padronizada com uma extensão de sistemas de transmissão de vídeo e TV digital (DVB). Os protocolos definem como podem ser construídas as camadas físicas e MAC de modo a prover uma maneira eficiente de transformar uma rede de transmissão via satélite em uma solução *Very Small Aperture Terminal* (VSAT), a qual possui a capacidade de realizar o transporte de IP em sistemas baseados em satélite. Ademais, as normas descrevem as interfaces usadas entre a HUB e o operador de satélite e o terminal interativo do usuário, para detalhar os processos que envolvem a construção de um protocolo capaz de realizar a comunicação com uma rede satelital.

As especificações do DVB-RCS2 descrevem componentes de camadas superiores adaptadas para fornecer sistemas interativos via satélite. Esses componentes são dependentes do padrão DVB. A rede DVB-RSC2 utiliza um satélite que funciona com uma cobertura de feixe único ou múltiplo, de forma que na maioria das redes, o satélite funciona tanto para transporte do link direto, como para o link de retorno.

O link direto carrega as informações relativas à sinalização do NCC bem como o tráfego de usuários para os RCSTs. A sinalização do NCC para os RCSTs é fundamental para operar o sistema de link de retorno, a qual é denominada Sinalização de link de encaminhamento (ETSI, 2014-04).

A arquitetura de codificação do protocolo DVB-RCS2 apresentada na figura 1 é composta pela junção de sub-módulos. Dentre esses submódulos pode-se destacar os blocos *RCST Controller*, *RLE Transmitter*, *GSE Receiver*. Cada um desses módulos é descrito pela norma DVB-RCS2. As informações são recebidas pelo link direto e são enviadas a um decodificador GSE, após a decodificação os dados são enviados a outros módulos do sistema, dentre esses o módulo *Rcs2 Controller* é responsável por gerenciar todo o funcionamento do terminal. O *Rcs2 Controller* consiste no MF-TDMA. Este módulo reage às informações recebidas da HUB, de forma a receber sinais de controle, com tabelas e descritores contendo os dados para a correta codificação do terminal. A codificação é realizada pelo bloco *RLE transmitter* a qual depende de informações repassadas pelo *rcst controller*. Após a codificação o sinal é encaminhado ao canal de retorno (RL).

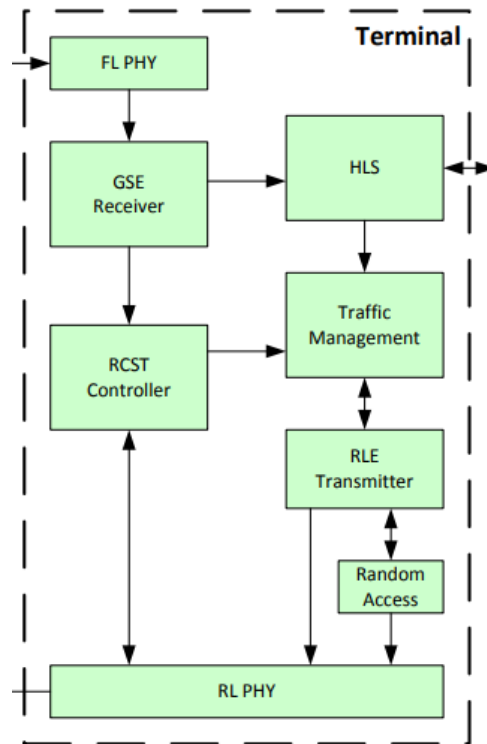


Figura 1 – Arquitetura básica do Terminal DVB-RCS2. Retirado de (ERL; COLA, 2014)

2.2 Moduladores

A norma DVB-RCS2 apresenta dois tipos básicos de modulação suportadas pelo *Return Channel over Satellite Terminal* (RCST), sendo essas a *linear modulation for the turbo coded* (TC-LM) e *Spread-Spectrum Linear Modulation* (TC-SSLM). O diagrama de blocos da figura 2 apresenta o fluxograma do sistema codificador, onde são verificados os módulos relativos a *Un-spread* (TC-LM) e *Spread* (TC-SSLM). O módulo de modulação recebe o *burst* mapeado em símbolos. Este bloco é composto internamente por outros 2 blocos:

- *Sobre-amostrador (upsampling)*.
- *Filtro raiz quadrada do cosseno levantado (SRRC)*.

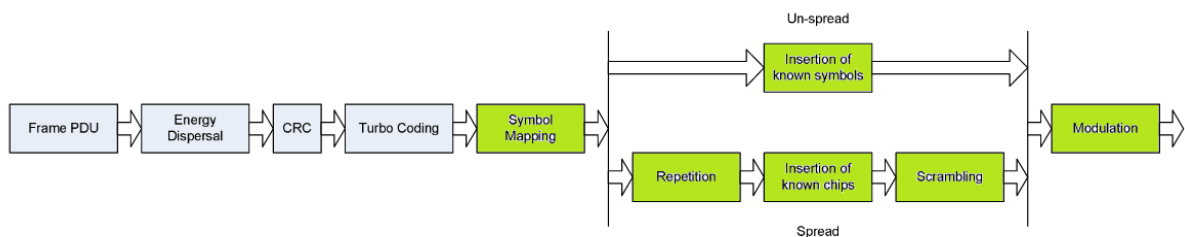


Figura 2 – Blocos do Codificador DVB-RCS2. Retirado de (ERL; COLA, 2014)

2.2.1 Sobre-amostrador

Este módulo trabalha em conjunto com o filtro SRRC para minimizar o efeito causado pelo pulso digital no domínio da frequência. Sua operação consiste em adicionar L zeros entre amostras recebidas.

$$Y_{out}[n] = X_{in}[n], n = 0, 1, \dots, L - 1 \quad (2.1)$$

$$Y_{out}[n] = 0, n = L, L + 1, \dots, N - 1 \quad (2.2)$$

2.2.2 Filtro raiz quadrada do cosseno levantado

O filtro SRRC, tem a finalidade de adequar o sinal codificado para satisfazer o critério de *Nyquist*, de forma que o sinal de saída esteja livre de ISI. Este filtro é necessário, pois um sinal digital consiste em um pulso, o qual no domínio da frequência é visto como um sinal periódico, logo uma sequência de pulsos gera uma interferência no domínio da frequência entre as amostras do sinal. O filtro SRRC desloca o sinal de forma que a região de maior energia esteja coincidindo com a posição em que os demais sinais não possuam energia. A figura 3 apresenta um exemplo da resposta ao impulso de um filtro SRRC.

O filtro SRRC possui a sua equação teórica definida pelas seguintes expressões matemáticas.

$$H(f) = 1 \quad \wedge \quad |f| < f_N(1 - \alpha) \quad (2.3)$$

$$H(f) = \sqrt{\frac{1}{2} + \frac{1}{2} \sin \frac{\pi(f_N - |f|)}{2\alpha f_N}} \quad \wedge \quad f_N(1 - \alpha) \leq |f| \leq f_N(1 + \alpha) \quad (2.4)$$

$$H(f) = 0 \quad \wedge \quad |f| > f_N(1 + \alpha) \quad (2.5)$$

onde $F_N = \frac{1}{2T_s} = \frac{R_s}{2}$, e o α consiste no valor de *roll-off* do filtro.

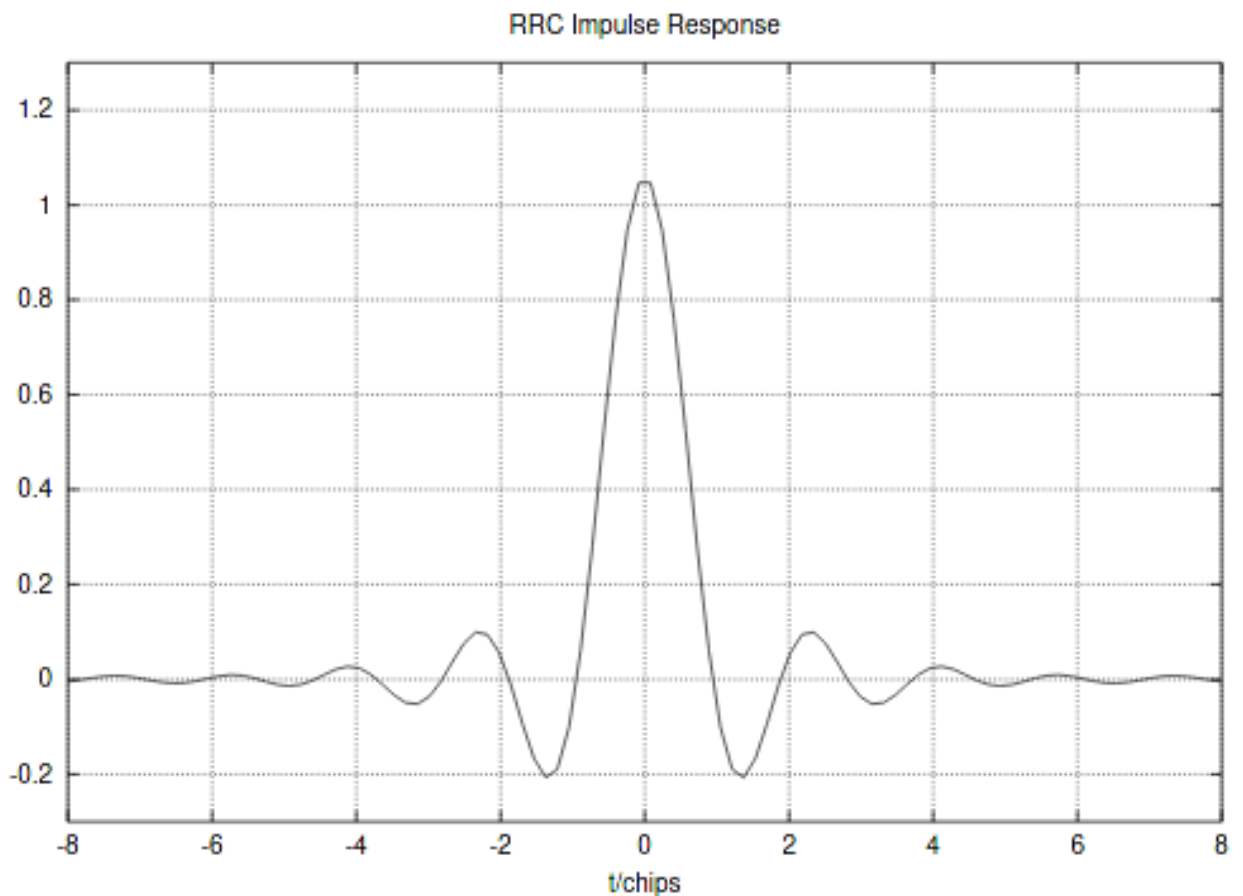


Figura 3 – Resposta ao impulso do filtro SRRC retirado de (JOOST, 2010)

2.2.3 Esquemas de modulação

O protocolo DVB-RCS2 determina que para os sistemas TC-SSLM e TC-LM, os esquemas de modulações sejam os seguintes.

- $\pi/2$ -BPSK
- QPSK
- 8PSK
- 16QAM

Os símbolos são mapeados de acordo com um conjunto de regras definidos no protocolo. Cada formato de modulação possui o seu próprio esquema de mapeamento de bit para símbolo.

2.2.4 $\pi/2$ -BPSK

O valor de u_0 apresentado na figura 4 é primeiro símbolo BPSK mapeado com +1 representando um bit de saída do codificador '0' e -1 representa um bit de saída do

codificador '1' conforme a equação 2.6. Todos os símbolos no quadro são posteriormente mapeados para modulação Pi/2-BPSK. Este mapeamento é feito pela rotação de fase externa. As figuras 4 e 5 mostram respectivamente o mapeamento simbólico e o mapeamento da constelação desta modulação. Onde é possível observar que o mapeamento é feito em pares S0,S2 e S3 e S1 em cada instante de tempo. O codificador de símbolos utiliza a seguinte equação. De forma que $u(n)$ é definido a segunda coluna da figura 4. Os símbolos A_0, B_0, A_1, B_1 representam os bits de dados a serem mapeados de bits para símbolos. A cada instante de tempo os símbolos sofrem uma rotação, o que gera uma constelação com um formato de QPSK.

$$s(n) = u(n)e^{(j\pi n/2 + j\pi/4)} \tag{2.6}$$

| Symbol index | u_0 |
|--------------|---------------|
| 0 | A_0 |
| 1 | B_0 |
| 2 | A_1 |
| 3 | B_1 |
| ... | |
| N-2 | $A_{N/2-1}$ |
| N-1 | $B_{N/2-1}$ |
| N | $Z_{1,0}$ |
| N+1 | $Z_{2,0}$ |
| N+2 | $Z_{1,1}$ |
| N+3 | $Z_{2,1}$ |
| ... | |
| N+M-2 | $Z_{1,M/2-1}$ |
| N+M-1 | $Z_{2,M/2-1}$ |

Figura 4 – Mapeamento de bit para símbolo: Modulação $\pi/2$ -BPSK.(ETSI, 2014-04)

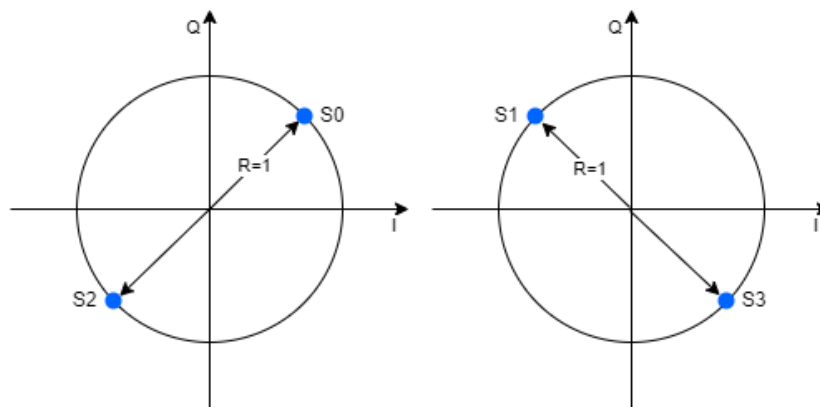


Figura 5 – Constelação: Modulação $\pi/2$ -BPSK(ETSI, 2014-04)

2.2.5 QPSK

Para os casos onde a modulação QPSK for utilizada, os bits sistêmicos (A , B) são transmitidos primeiro, seguidos dos bits de paridade (Z_1, Z_2), tal como apresentado na figura 6. A figura 7 mostra o mapeamento da constelação QPSK.

| Symbol index | u_0 | u_1 |
|--------------|-----------|-----------|
| 0 | A_0 | B_0 |
| 1 | A_1 | B_1 |
| ... | | |
| $N-1$ | A_{N-1} | B_{N-1} |
| N | $Z_{1,0}$ | $Z_{2,0}$ |
| $N+1$ | $Z_{1,1}$ | $Z_{2,1}$ |
| ... | | |
| $N+M$ | $Z_{1,M}$ | $Z_{2,M}$ |

Figura 6 – Mapeamento de bit para símbolo: Modulação QPSK.(ETSI, 2014-04)

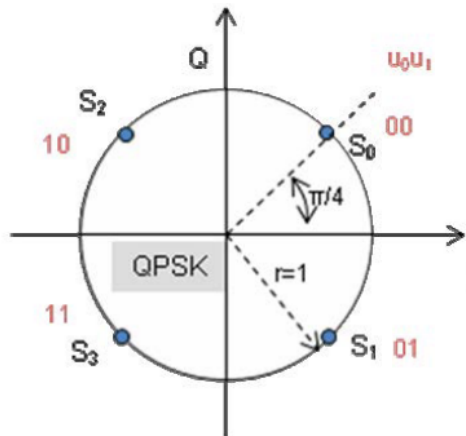


Figura 7 – Constelação: Modulação QPSK.(ETSI, 2014-04)

2.2.6 8PSK

Os bits sistêmicos(A, B) são transmitidos primeiro, seguidos do envio dos bits de paridade (Z_1, Z_2). O mapeamento dos símbolos depende da taxa de código usada, tal como apresentado nas figuras 8 e 9.

| Symbol index | u_0 | u_1 | u_2 |
|--------------|-------------|------------|------------|
| 0 | $Z_{1,0}$ | A_0 | B_0 |
| 1 | $Z_{2,0}$ | A_1 | B_1 |
| 2 | $Z_{1,1}$ | A_2 | B_2 |
| 3 | $Z_{2,1}$ | A_3 | B_3 |
| ... | | | |
| 2N-2 | $Z_{1,N-1}$ | A_{2N-2} | B_{2N-2} |
| 2N-1 | $Z_{2,N-1}$ | A_{2N-1} | B_{2N-1} |

Figura 8 – Mapeamento de bit para símbolo: Modulação 8PSK - rate 2/3.(ETSI, 2014-04)

| Symbol index | u_0 | u_1 | u_2 |
|------------------------------------|-----------|-----------|-----------|
| 0 | $Z_{1,0}$ | A_0 | B_0 |
| 1 | $Z_{2,0}$ | A_1 | B_1 |
| 2 | $Z_{1,1}$ | A_2 | B_2 |
| 3 | $Z_{2,1}$ | A_3 | B_3 |
| ... | | | |
| 2k | $Z_{1,k}$ | A_k | B_k |
| 2k+1 | $Z_{2,k}$ | A_{k+1} | B_{k+1} |
| When all 2M parity bits are given: | | | |
| 2M | A_M | B_M | A_{M+1} |
| 2M+1 | B_{M+1} | A_{M+2} | B_{M+2} |
| ... | | | |

Figura 9 – Mapeamento de bit para símbolo: Modulação 8PSK - rate 5/6 e 3/4.(ETSI, 2014-04)

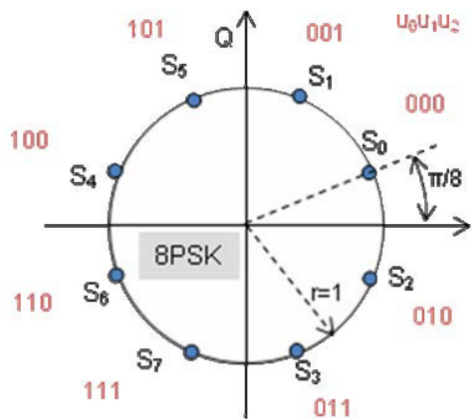


Figura 10 – Constelação: Modulação 8-BPSK.(ETSI, 2014-04)

2.2.7 16QAM

O mapeamento em um sistema 16-QAM, consiste em decompor o sistema em 4 modulações por amplitude de pulso (4-PAM), de forma que os eixos da constelação serão (I-PAM E Q-PAM) para fase e quadratura respectivamente. O Sistema de 16-QAM é construído para operar em taxas de códigos maiores ou iguais a $3/4$.

Para taxas maiores a $3/4$, o número de símbolos será maior que o número de Z-bits no frame. Os Z-bits devem ser direcionados aos primeiros $2M$ símbolos, e o resto dos bits A e B são usados para preencher o resto dos símbolos, na ordem que deixam o codificador, como retratado na figura 11. (ETSI, 2014-04).

| Symbol index | Q-PAM | | I-PAM | |
|---|-------------|-------------|---------------|---------------|
| | u_{Q1} | u_{Q0} | u_{I1} | u_{I0} |
| 0 | $Z_{1,0}$ | A_0 | B_0 | A_1 |
| 1 | $Z_{2,0}$ | B_1 | A_2 | B_2 |
| 2 | $Z_{1,1}$ | A_3 | B_3 | A_4 |
| 3 | $Z_{2,1}$ | B_4 | A_5 | B_5 |
| ... | | | | |
| $2k$ | $Z_{1,k}$ | A_{3k} | B_{3k} | A_{3k+1} |
| $2k+1$ | $Z_{2,k}$ | B_{3k+1} | A_{3k+2} | B_{3k+2} |
| When all (2M) parity bits are given and code rate > $3/4$ | | | | |
| $2M$ | A_{3M} | B_{3M} | A_{3M+1} | B_{3M+1} |
| ... | | | | |
| $2M+k$ | A_{3M+2k} | B_{3M+2k} | $A_{3M+2k+1}$ | $B_{3M+2k+1}$ |
| ... | | | | |

Figura 11 – Mapeamento de bit para símbolo: Modulação 16-QAM.(ETSI, 2014-04)

| u_0u_1 | 4-PAM value |
|----------|----------------|
| 00 | $-1/\sqrt{10}$ |
| 01 | $+1/\sqrt{10}$ |
| 10 | $-3/\sqrt{10}$ |
| 11 | $+3/\sqrt{10}$ |

Figura 12 – Mapeamento da constelação: Modulação 16-QAM(ETSI, 2014-04)

O bloco de modulação foi implementado em *hardware* anteriormente em (SANTOS, 2019). Neste trabalho, foram implementadas as 4 modulações $\pi/2$ BPSK, QPSK, 8PSK e 16QAM, com os blocos de *upsampling* e filtro SRRC. O sistema é implementado em ponto fixo e apresenta os resultados obtidos na saída do filtro, onde é feita a validação do sistema a partir e modelos implementados em octave. O trabalho avançou até a etapa de filtragem do sinal, sendo ainda necessário a implementação do bloco *mixer*, o qual é responsável por realizar a multiplicação do sinal por uma frequência desejada. O presente trabalho tem como um dos objetivos validar esta a implementação física com a solução implementada em ARM e com os modelos em python criados do sistema.

2.3 Espalhamento de espectro

A modulação por espalhamento de sinal (*Spread Spectrum Linear Modulation*), resulta em um sinal em que sua banda de frequências é bem maior que a necessária para realizar a transmissão. Os sinais modulados utilizando essa técnica, são menos suscetíveis a sofrerem interferências (CS457/CS546... , 2001).

2.3.1 Banda estreita

Sinais em banda estreita possuem a maioria da sua potência concentrada em uma determinada região da banda de passagem de frequências. Desta forma, estes sinais são suscetíveis a interferência, e podem ser detectados por outros sistemas (CS457/CS546... , 2001).

2.3.2 Gerador de sequências

A arquitetura de um sistema que utiliza o espalhamento de espectro pode ser observada na seguinte figura 13. A função $c(t)$ é responsável por realizar o embaralhamento do sinal. De forma geral, esta função consiste em um gerador de números aleatórios. O qual pode ser construídos a partir de operações lógicas e registradores (NETTO, 2009).

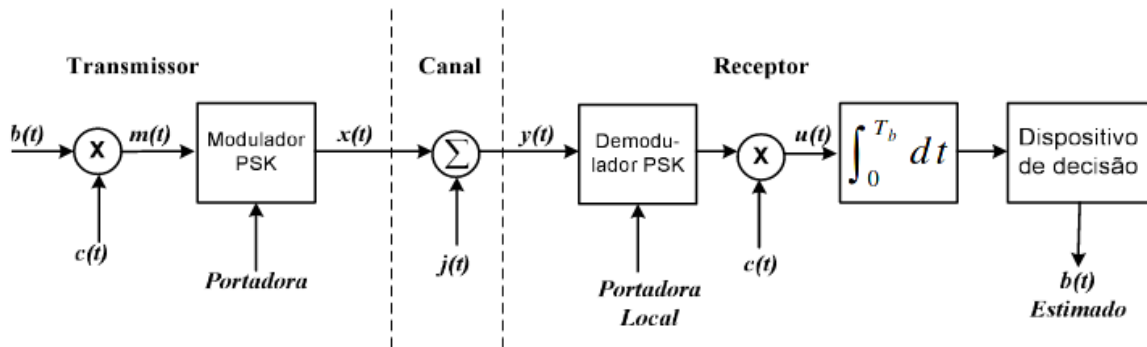


Figura 13 – Mapeamento da constelação: Modulação 16-QAM (NETTO, 2009).

2.3.3 Espalhamento das frequências

Esta técnica consiste em espalhar a banda estreita em uma faixa maior de frequências, desta forma a potência do sinal não estará concentrada em uma faixa estreita, possibilitando que o sinal se torne mais robusto, sendo menos suscetível a ruídos (CS457/CS546... , 2001).

Os sistemas que utilizam modulações com espalhamento de frequências possuem as seguintes vantagens em relação a sistemas que utilizam banda estreita (WEI, 2016).

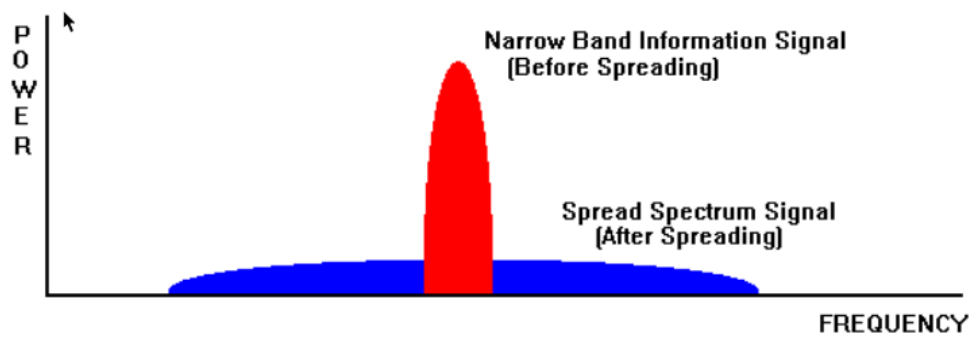


Figura 14 – Sinal em banda estreita e com espalhamento (CS457/CS546. . . , 2001).

- Densidade espectral baixa: O fato de o sinal estar espalhado pelas frequências, faz com que a sua densidade espectral de potência fique menor.
- Códigos aleatórios: O sinal sofre um embaralhamento a partir de um pseudo código, o qual faz que o sinal fique mais difícil de ser detectado por outros sistemas.
- Acesso a qualquer momento: O usuário consegue iniciar a transmissão a qualquer intervalo de tempo

2.3.4 Sistema utilizado no protocolo DVB-RCS2

O protocolo DVB-RCS2 apresenta um processo de espalhamento de frequências (*spread-spectrum*) construído pelo polinômio (2.7), o qual é utilizado para gerar números aleatórios de 1 e 0, os quais são traduzidos para -1 e 1.

$$1 + X + X^6 + X^{10} + X^{14} \tag{2.7}$$

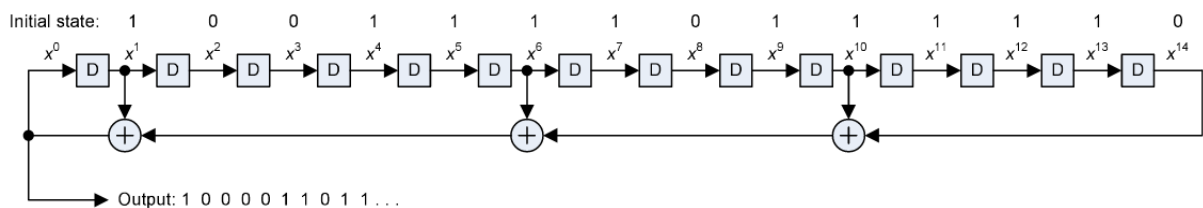


Figura 15 – Exemplo para geração de sequência de embaralhamento (ETSI, 2014-04).

Os valores obtidos na saída do gerador de sequências são multiplicados por cada símbolo de entrada. O fluxograma do processo para obter os valores de *spread* são apresentados na figura 15.

2.4 MF-TDMA

As técnicas envolvidas no sistema *Multi-Frequency-Time-Division* (MF-TDMA) permitem que um grande número de usuários compartilhem uma banda com eficiência. O sistema permite que os recursos da rede, sejam alocados conforme a necessidade e demandas do usuário. Os recursos alocados para usuários individuais, devem ser agrupados no espectro disponível para o sistema (YAZDANI, 2008).

Em sistemas de telecomunicações os recursos são limitados. Desta forma, quando muitos usuários fazem acesso a uma rede, é necessário arbitrar a utilização desses recursos entre os usuários (YAZDANI, 2008). O sistema de acesso por divisão de tempo, ou TDMA divide o tempo que cada usuário pode acessar a largura de banda total. Desta forma, o recurso é arbitrado pelo compartilhamento do tempo (YAZDANI, 2008).

O sistema de acesso por divisão de tempo multi frequencial (MF-TDMA) divide os usuários em regiões de frequência e tempo. Cada usuário recebe um conjunto de sub-larguras de banda em um determinado tempo, como pode ser observado na figura 17.

2.4.1 Super Frame

O *super frame* é composto por *frames* constituídos por *timeslots*, os quais são limitados no tempo e na frequência, como mostrado na figura 16. Dentro do *super frame*, os frames são numerados a partir da frequência mais baixa, primeiro no tempo, e menor *frame type*. Entre um *super frame* e outro, é inserido um tempo de guarda, para evitar erros durante a recepção de um conjunto de *super frames*. (ETSI, 2014-04).

O protocolo DVB-RCS2 determina que a duração de um *super frame* seja de 25 ms a 750 ms.

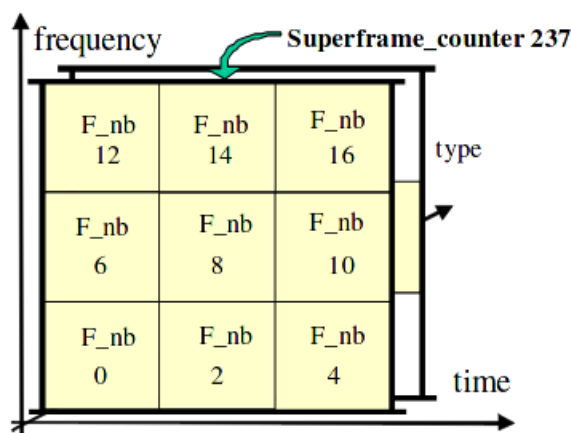


Figura 16 – Super Frame - Frames disponíveis (ETSI, 2014-04)

2.4.2 Frame

O *frame* é composto por um conjunto de *timeslots*, cada *frame* possui um tipo de transmissão associada (vide figura 17). O *super frame* é composto por unidades chamadas de *bandwidth-Time Units* (BTUs), a partir dessas unidades básicas, é possível a construção de *timeslots*, os quais formaram o *frame* (ETSI, 2014-04).

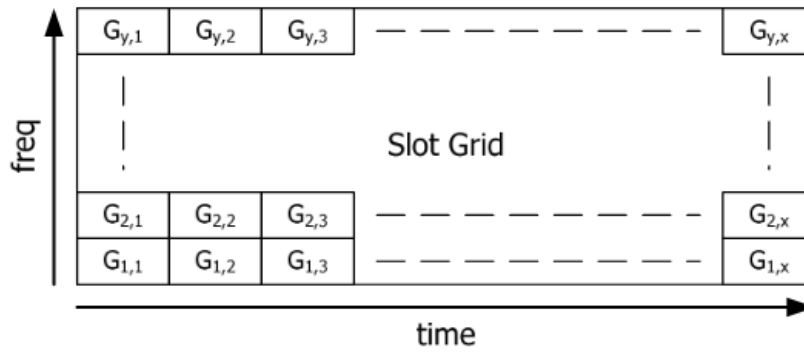


Figura 17 – Frame: Slots disponíveis(ETSI, 2014-04)

2.4.3 Timeslot

O *timeslot* é composto por BTUs, esses BTUs são organizados dentro do *timeslot* de forma que uma parcela do tempo dos BTUs será utilizada como um tempo de guarda do sistema.



Figura 18 – Timeslots e BTUs(ETSI, 2014-04)

2.4.4 Tempo de guarda

A figura 18 apresenta um *timeslot*, onde é possível verificar que o começo dos dados do *timeslot* inicia-se com um tempo de folga, isto se deve ao fator de o *timeslot* possuir um tempo guarda em seu início e fim. Este tempo é utilizado para minimizar erros entre transmissões consecutivas.

2.4.5 Sincronização de relógio

O sincronismo de relógio é fundamental para o correto funcionamento de um sistema que envolve modulação. Desta forma, o *clock* do transmissor e do receptor devem ser sempre sincronizados, para que erros de demodulação e recepção de dados sejam evitados.

Os sistemas que utilizam a norma DVB-RCS2 utilizam o sistema de PCR, de forma que um valor de referência é enviado em intervalos de 40ms ao receptor. A partir deste valor, o sistema deve conseguir realizar a sincronização de relógio (ETSI, 2014-04).

2.4.6 Malha de captura de fase

O circuito PLL ou *Phase Locked Loop* é um importante circuito que se encontra em diversas aplicações na área de eletrônica. Este tipo de circuito é muito utilizado em receptores AM, FM, modems, sintetizadores de frequências, telefones celulares, instrumentos digitais e analógicos, entre outras aplicações que trabalham com frequência (LATA; KUMAR, 2013). A seguir é apresentada uma lista dos sistemas PLL utilizados em aplicações da área de eletrônica.

- PLL linear(LPLL)
- PLL digital(DPLL)
- PLL em software(SPLL)
- PLL com entrada digital(ADPLL)

O PLL funciona como um circuito capaz de rastrear a frequência e fase de um sinal de entrada a partir de um comparador que recebe como parâmetros um sinal a ser rastreado, bem como uma realimentação advinda de um circuito controlado por tensão. O resultado da comparação desses sinais consiste no erro entre esses sinais, este erro, é inserido em um filtro, geralmente passa baixas, para eliminar componentes de altas frequências, as quais estão ligadas principalmente a componentes de *jitter* da rede(LATA; KUMAR, 2013).

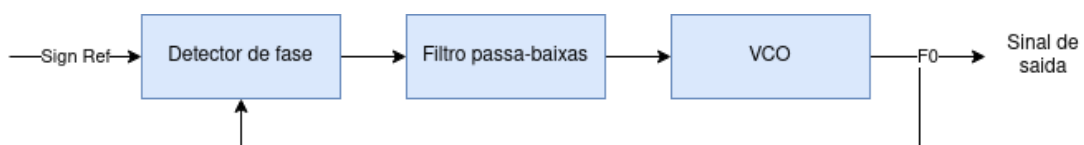


Figura 19 – Diagrama de um sistema PLL

- Comparador / Detector de fase: Compara 2 sinais, e obtém a fase, a partir deste erro gera uma voltagem relativa a esta diferença.

- Filtro: Remove componentes indesejados do sistemas, como ruídos. A filtragem é realizada após a diferença de fase entre os sinais.
- Oscilador controlado por tensão: Gera a frequência de saída do sistema a partir de um sinal de entrada, e é considerada a saída do circuito.

A chave do funcionamento do circuito PLL se baseia na habilidade de conseguir detectar a diferença de fase entre 2 sinais. A partir desta diferença de fase, o sistema consegue realizar o controle da frequência do sistema, para reduzir este erro (LATA; KUMAR, 2013).

2.5 Hardware Reconfigurável e Sistemas em Chip (SoC)

A figura 20 representa a arquitetura interna de uma FPGA (*Field Programmable Gate Array*), que é um chip que possibilita a programação a nível de portas lógicas.

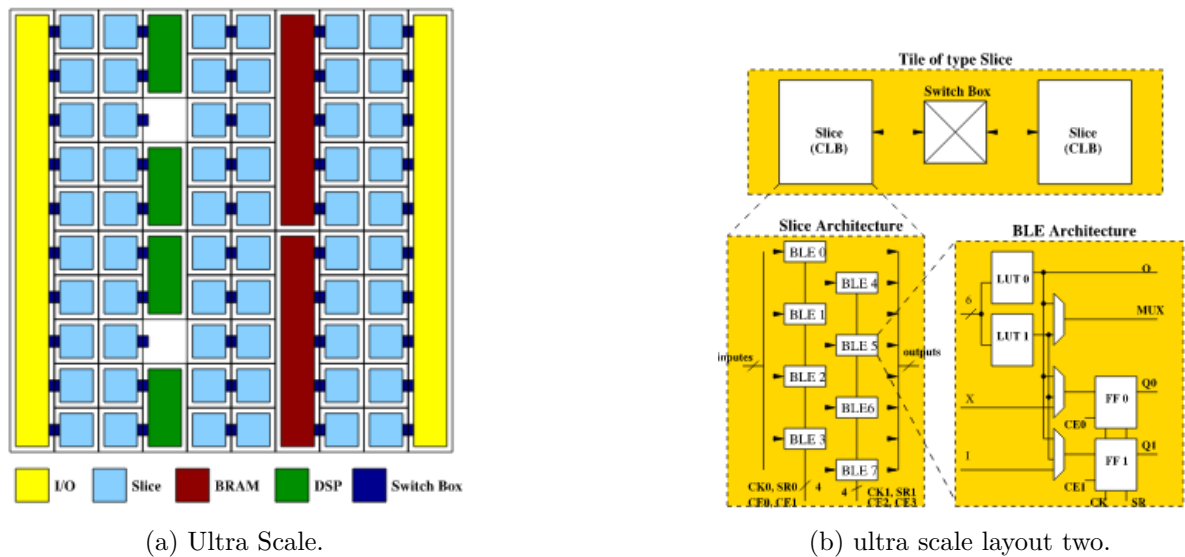


Figura 20 – Arquitetura geral de uma FPGA Zynq Ultrascale (ABUOWAIMER DANI MAAROUF; VANNELLI, 2018)

Uma FPGA é formada por uma matriz de blocos lógicos programáveis, constituídos por slices, que por sua vez são formados por algumas *Look-Up tables (LUTs)*, multiplexadores e registradores (latches ou flip-flops). As LUTs permitem a descrição de portas lógicas ou circuitos combinacionais, enquanto os registradores permitem armazenar o estado do circuito implementando lógica sequencial. Os blocos lógicos programáveis podem ser conectados através de uma rede de roteamento baseada em memórias SRAM para implementar circuitos mais complexos. Adicionalmente, FPGAs contêm blocos de memória (BRAMs), blocos para processamento digital (DSPs), entre outros circuitos dedicados como conversores de dados ADC, PLLs e circuitos de geração de clock. Os *slices* possuem

tanto LUTs como *Flip-Flops* (FFs), que quando são agrupados juntos, compartilham o mesmo *switch* para roteamento (ABUOWAIMER DANI MAAROUF; VANNELLI, 2018).

Observa-se ainda que os *slices* são colocados lado a lado, para compartilharem o mesmo bloco de *switches*. Cada *slice* contém 8 elementos de lógica básica (BLEs), 2 LUT com 6 entradas, muxes e 2 FFs (ABUOWAIMER DANI MAAROUF; VANNELLI, 2018).

A programação de uma FPGA utiliza uma linguagem de descrição de *hardware*, sendo que na indústria os que são mais utilizadas são o VHDL e o Verilog. A partir desta linguagem é possível construir sistemas de alta complexidade a partir da descrição lógica de circuitos combinacionais e sequenciais.

O desenvolvimento do sistema embarcados de alto nível, permitiram que as plataformas de desenvolvimento de um projeto em *hardware*, tenham a capacidade de se integrar com processadores de propósito geral, como o processador ARM, constituindo Sistema em Chip (SoCs). A utilização desses sistemas oferecem uma grande flexibilidade ao projetista, tendo em vista a capacidade que os mesmos possuem de realizar tarefas de alto nível, a partir de instruções customizadas.

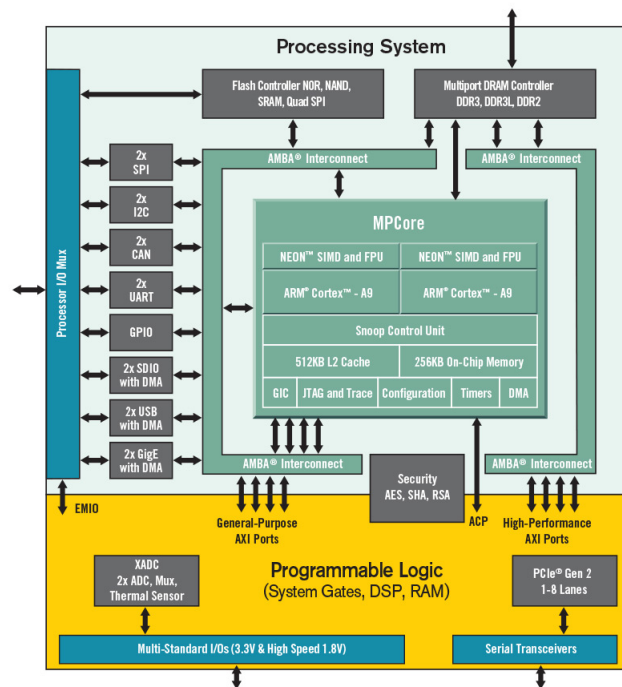


Figura 21 – SoC FPGA Zynq-7000 (XILINX,)

A figura 21 apresenta a arquitetura geral do *System of Chip FPGA Zynq-7000*. Os blocos de programação em hardware do sistema apresenta DSP, RAM, portas lógicas e barramentos do tipo AXI. O sistema apresenta um processador dual core ARM Cortex-A9 com uma memória L2 cache de 512 KBs, e uma memória On-Chip de 256 KBs.

2.6 Estado da Arte

A seguir será apresentada alguns trabalhos utilizados como referência para o desenvolvimento deste trabalho.

2.6.1 Filtro SRRC

O filtro *Root Raised Cosine* (SRRC) utilizado no módulo da modulação já foi implementado em FPGA pela comunidade científica. As implementações podem servir como referência, e em muitas vezes diferem em técnicas e linguagem de programação utilizadas. Em (SATEESHKUMAR, 2013) é proposta a arquitetura de um filtro SRRC baseado na norma (ETSI, 2014-04). A figura 22 mostra a arquitetura do filtro proposta neste trabalho.

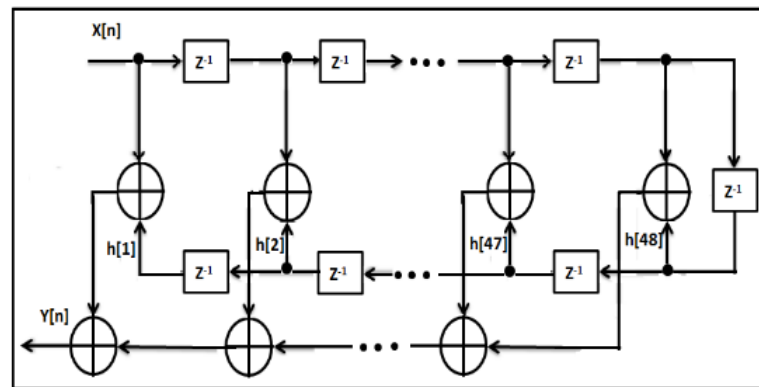


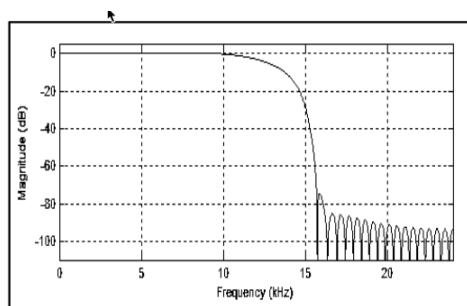
Figura 22 – Arquitetura do filtro RRC (SATEESHKUMAR, 2013)

São utilizados, multiplicadores, somadores e elementos de atraso para a sua construção. Os coeficientes $h[n]$ são dados como uma das entradas dos multiplicadores, nos quais recebem também uma entrada $x[n]$. Desta forma, o sistema utiliza uma FIFO, em que os dados entram no sistema, e a cada nova amostra na entrada gera em um certo momento o dado filtrado $y[n]$ na saída do sistema.

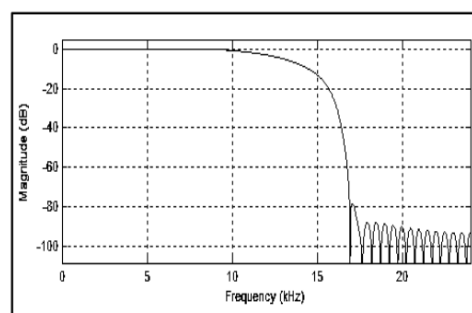
O trabalho utilizou o software ALTERA Quartus II e um FPGA Cyclone III. O software da altera é utilizado para realizar e otimizar o design do sistema a partir do manejo das células lógicas e das próprias conexões entre elas (SATEESHKUMAR, 2013).

Os resultados obtidos no sistema são mostrados na figura 23, de forma que são considerados 2 casos de teste para valores de *Roll-Off* de 0.25 e 0.35

Os resultados obtidos demonstraram que a implementação proposta pode ser utilizada em sistemas DVB-RCS2, tendo em vista que o mesmo filtro é utilizado neste sistema.



(a) Magnitude do filtro com Roll-off = 0.25



(b) Magnitude do filtro com Roll-off = 0.35

Figura 23 – Magnitude do filtro RRC (SATEESHKUMAR, 2013)

2.6.2 MF-TDMA

Em (JEONG JOONGYU RYU, 2003) foi implementado o MF-TDMA utilizando uma arquitetura um pouco diferente da apresentada pela norma (ETSI, 2014-04). A figura 24 mostra a arquitetura proposta.

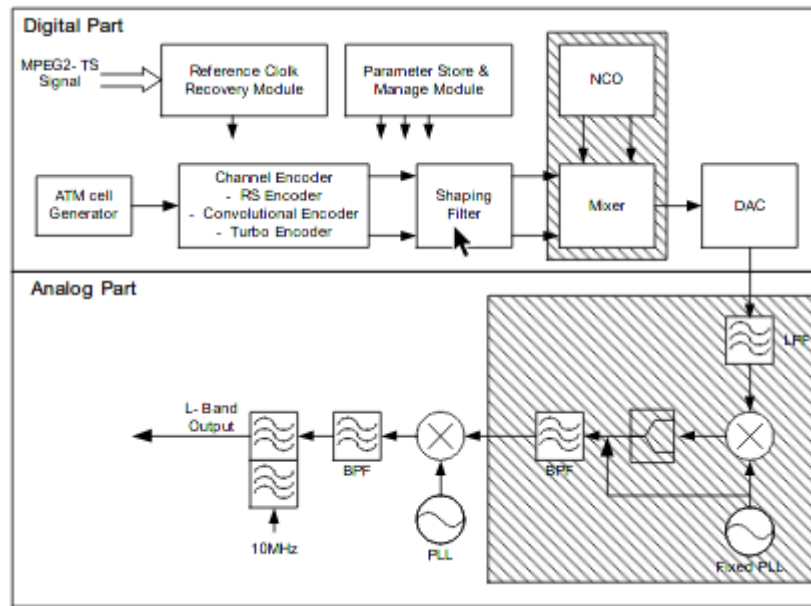


Figura 24 – Arquitetura geral I do MF-TDMA proposta em (JEONG JOONGYU RYU, 2003)

O sistema demonstrou que a frequência de saída, pode estar em torno de 10 a 30 MHz, contudo, para sintetizar uma frequência de 30MHz o sistema deve operar com um *clock* maior que 90 MHz. Ademais, em sistemas com *clock* maior que 90MHz podem surgir problemas nos processos de multiplicação (JEONG JOONGYU RYU, 2003).

Também é apresentado em (JEONG JOONGYU RYU, 2003) uma outra arquitetura do sistema, o qual reduz a complexidade da parte digital. O diagrama de blocos da figura 25 sugere que o sistema formado pelo NCO e *Mixer* sejam transferidos para o bloco da parte analógica.

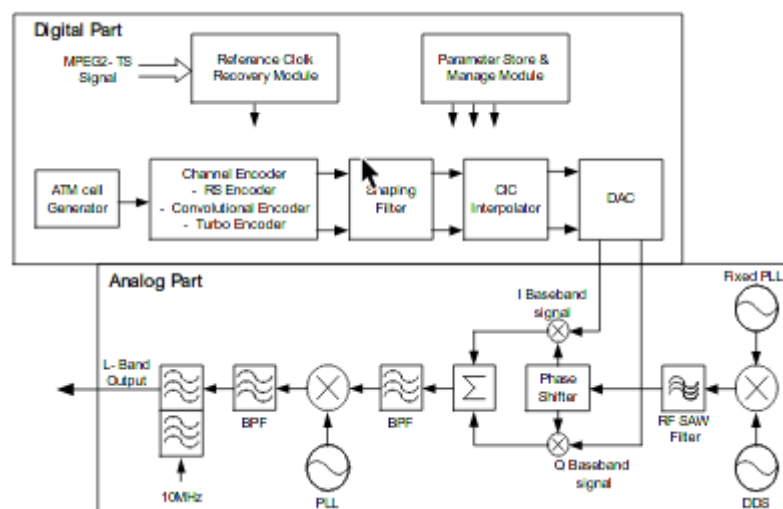


Figura 25 – Arquitetura geral II do MFTDMA proposta em (JEONG JOONGYU RYU, 2003)

Em (R.PRABHU R.NAGARAJAN, 2017) é construído um *Direct Sequence Spread*

Specrum (DSSS) utilizando FPGA. O sistema utiliza um gerador de ruído (PN), o qual possui certas propriedades de correlação. O diagrama de blocos do gerador de seqüências é construído como um sistema que utiliza 8 registradores, sendo 2 para cada estágio do sistema. O sistema é composto por 4 estágios, de forma que em cada estágio realiza a soma de um bit de cada um dos 2 registradores do estágio. Em seguida, este resultado é direcionado ao próximo estágio. Este processo é repetido até o último estágio, onde é feita a operação logica NOR e o resultado é então inserido no início do último registrador.

A arquitetura geral do sistema é apresentada, de tal forma que o sistema recebe os dados, e em sua saída retorna o valor do dado após a multiplicação pelo valor obtido do gerador de seqüências.

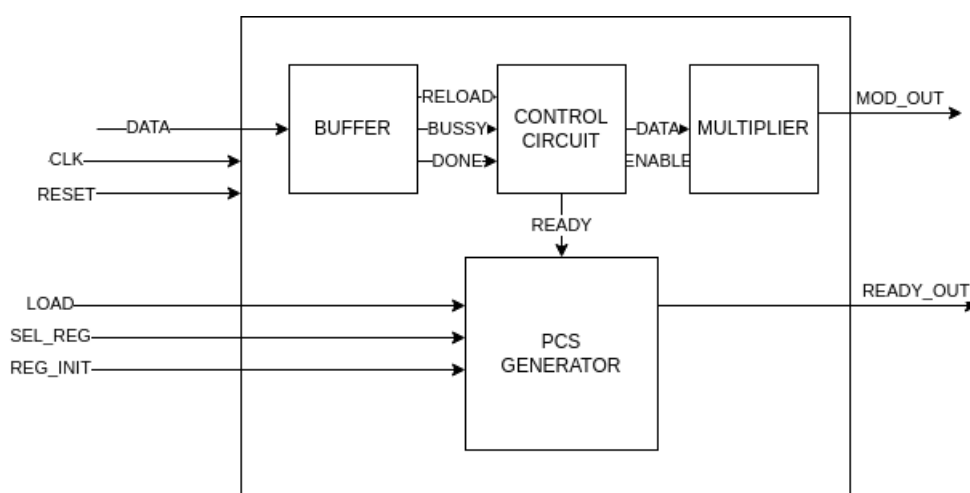


Figura 26 – Diagrama de blocos do DSSS proposta em (R.PRABHU R.NAGARAJAN, 2017)

Os resultados obtidos nesta implementação demonstraram que o gerador de seqüências pseudo aleatórias pode ser utilizado para gerar valores de saída a partir de uma semente (seed) como entrada. Os valores de saída do gerador, possuem uma certa correlação, as quais podem ser utilizadas no sistema (R.PRABHU R.NAGARAJAN, 2017).

2.6.3 CDMA

Em (SAROJINI, 2012) é sugerida a implementação de um *Code Divison Multiple Access* (CD-MA). O sistema utiliza uma seqüência pseudo aleatória (PN) a qual é modulada usando BPSK onde a portadora é gerada digitalmente a partir do conceito de frequência sintetizada. Os sinais modulados são combinados, e em seguida transmitidos, conforme apresentado na figura 27.

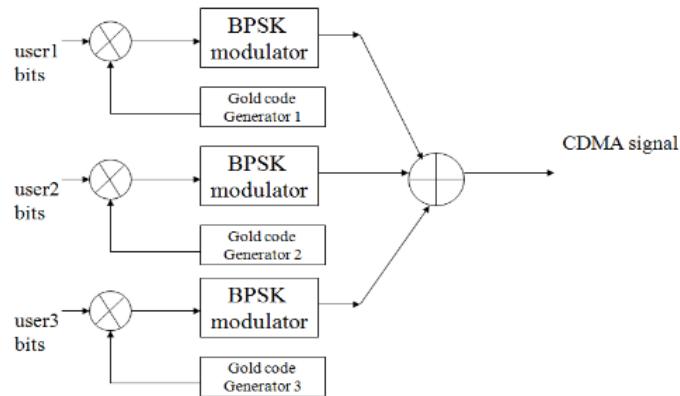


Figura 27 – Diagrama de blocos do CD-MA sugerida em (SAROJINI, 2012)

O sistema utiliza em sua construção um DDS para a geração das senoides, o qual utiliza os princípios dos PLLs para a sua implementação. A figura 28 apresenta a arquitetura geral do sistema.

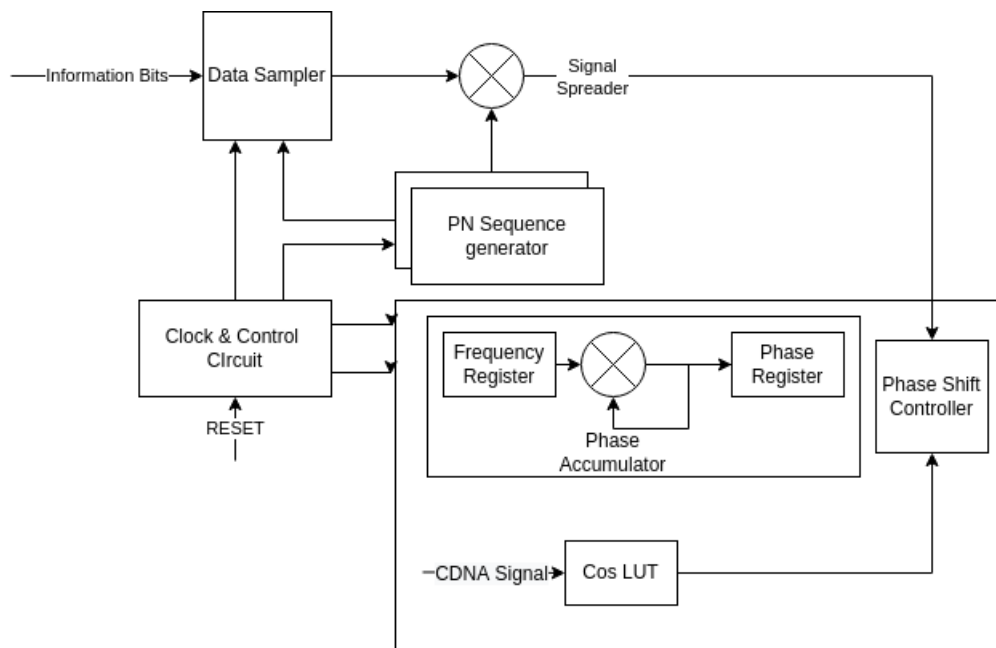


Figura 28 – Diagrama de blocos do CDMA sugerida em (SAROJINI, 2012)

O sistema proposto apresenta grande variedade de implementações de CD-MA. O sistema é totalmente reconfigurável em qualquer sistema de telecomunicação. O sistema utiliza 126 *gold code* seqüências, as quais podem ser geradas pelo pseudo gerador (PN) (SAROJINI, 2012).

As formas de onda apresentadas no trabalho demonstraram que o sinal de saída possui o mesmo tamanho que o sinal de entrada. Desta forma, é possível verificar a partir das formas de ondas obtidas da simulação do sistema que os dados recebidos são os mesmos que os dados dos usuários com a adição de algum delay.

2.6.4 Recuperação de sincronismo

Em (LISS, 2012) é sugerido o método para recuperação de sincronismo de relógio entre transmissor e receptor em sistemas DVB-RCS2. O projeto utiliza PLL's, com filtros adequadamente projetados para a eliminação de componentes de *jitter*. Na figura 29 é apresentado o digrama de blocos geral do sistema.

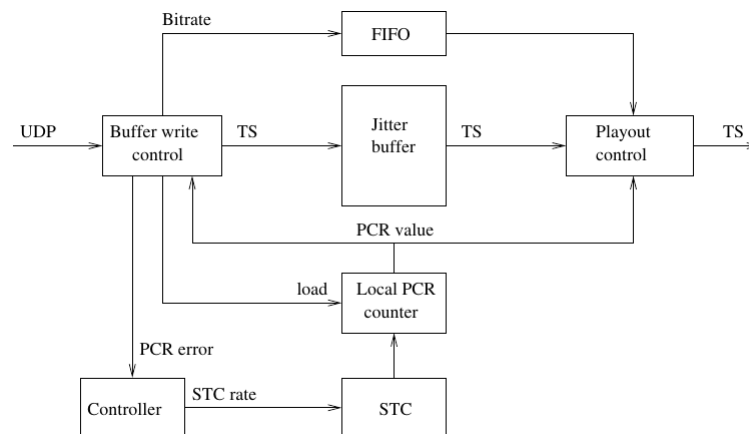


Figura 29 – Diagrama de blocos proposto em (LISS, 2012)

A implementação deste trabalho, utiliza o erro de fase entre o *Local PCR Counter* e o valor PCR recebido a partir do *Packet-Stream*. Este erro é enviado ao bloco *controller*, onde é feita a filtragem deste sinal de erro. A filtragem é necessária para eliminar as componentes de jitter presentes na rede. O valor filtrado é então utilizado para corrigir a frequência do *Local PCR Counter*. O resultado obtido nesta implementação é apresentado a na figura 30.

A figura superior, apresenta-se o intervalo de envio de cada *PCR packet*. A figura superior direita apresenta a distribuição de *jitter*, de forma que a barra do meio deve ser a mais significante. A figura inferior esquerda apresenta o valor PCR do sistema local, com uma margem de 40 ms, a qual é recomendada pelo sistema DVB. A figura inferior direita, apresenta os valores de PCR recebidos com *jitter*.

Em (JEON, 2014) é sugerida uma implementação para a recuperação dos *ticks* da HUB, a partir do uso de um sistema de re-alimentação em malha fechada. O sistema é composto por um PLL, onde esta presente um filtro IRR de primeira ordem, e um VCO para controlar a velocidade de contagem do contador local. A figura 31 apresenta a arquitetura proposta.

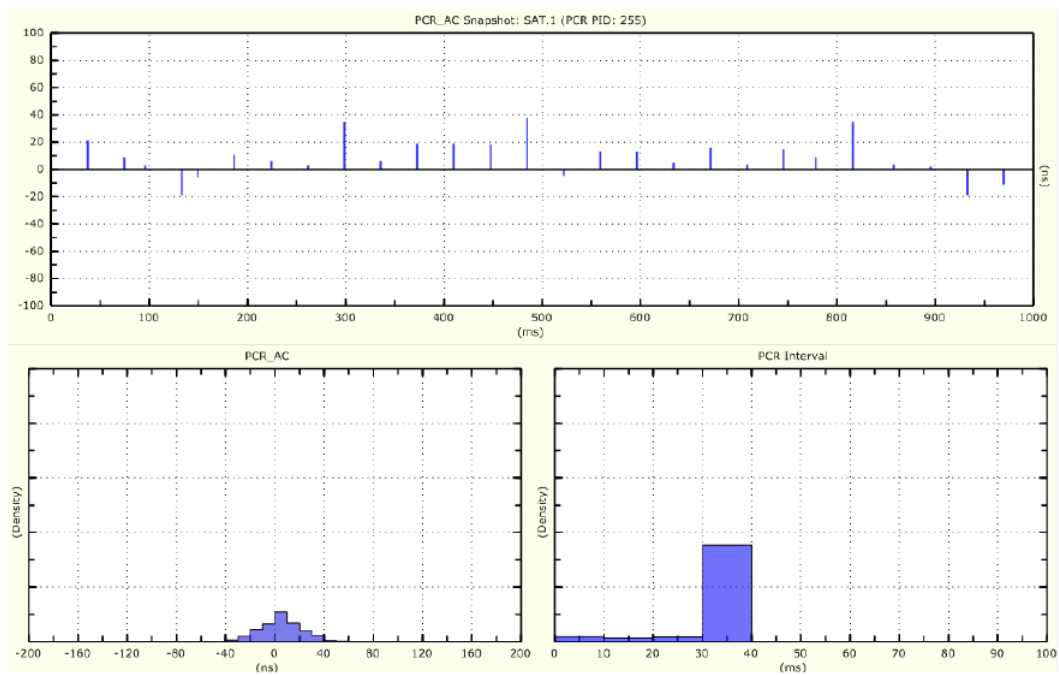


Figura 30 – Medições no tempo dos valores recebidos e corrigidos de PCR(LISS, 2012)

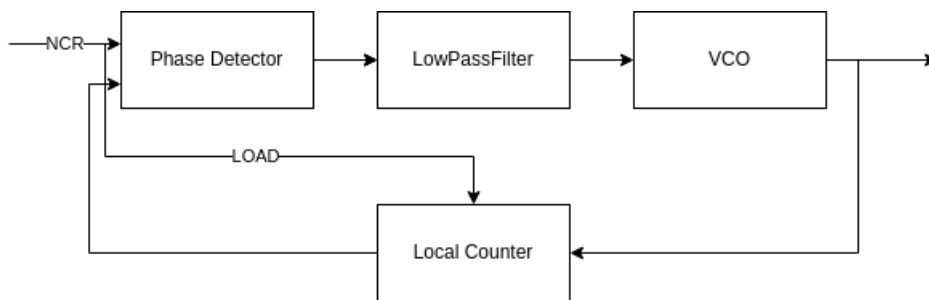


Figura 31 – Diagrama de blocos do sistema PLL para recuperação de relógio proposto em (JEON, 2014)

O sistema implementado em (JEON, 2014) recebe o valor de NCR da HUB, de forma que a norma DVB-RCS2 padrão sugere que o primeiro NCR deve ser obtido utilizando o SOF (*start of Frame*). A simulação do sistema é feita considerando algumas componentes de *jitter* presente na rede, e a alguns *delays* e ruídos no sistema.

Na tabela abaixo resume os detalhes de como foi implementado os módulos de modulação, *spread spectrun*, MFT-DMA e *NCR recovery*, seja a implementação em hardware, software ou SoC.

Tabela 1 – Detalhes da implementação de blocos do protocolo DVB-RCS2: Modulação, spread-spectrum e MF-TDMA.

| Paper | Algoritmos Utilizado | Tipo de implementação | objetivo | Consumo de recursos (HW) |
|------------------------------|---|-----------------------|--|---|
| (SATEESHKUMAR, 2013) | Convolução Linear PLL (mixer) Turbo Encoder (Codificação do sinal) | FPGA | Obter IS mla | Não Menciona |
| (JEONG, JOONGYU RYU, 2003) | Convolução Linear (Filtro RRC) | FPGA | Construção de um mfdma e uso em Chip | Não Menciona |
| (R.PRABHU R.NAGARAJAN, 2017) | Turbo Encoder (Codificação do sinal) algoritmo de gold (gerador de seqüência) | FPGA | Obter um gerador pseudo randômico | Não Menciona |
| (SAROJINI, 2012) | Algoritmo de gold (Gerador de seqüências) DDS (utiliza PLL) Modulador BPSK (modulação do sinal) | FPGA | Transmissor CDMA | LUT:2545 Flip Flops: 821 MULTI8X18SIOs: 1 GCLKs: 3 |
| (JEON, 2014) | Filtro de primeira ordem (Filtro IRR) PLL (sincronismo) | software | Recuperar sincronismo entre HUB e terminal em sistema DVB-RCS2 | Não Menciona |
| (LISS, 2012) | PPL(Sincronização de relógio) Filtragem Adaptativa (Eliminar Jitter) | FPGA | Obter sincronismo de relógio entre sistemas separados | Slice:550 Flip Flops: 1630 LUT: 1750 SRL : 88 DSPs : 13 |

3 Metodologia e ferramentas

Para a correta implementação dos módulos de codificação *modulation*, *spreading* e do *MF-TDMA* diversos trabalhos foram avaliados em termos do algoritmo ou técnica usada, tempo de execução e complexidade envolvida nesses projetos. Neste trabalho, o módulo de *spreading* e o MF-TDMA foram baseados principalmente no protocolo DVB-RCS2 (ETSI, 2014-04).

3.1 Codificador RCS2

3.1.1 Esquemas de modulação

A Modulação utilizada no sistema é feita utilizando os esquemas QPSK, BPSK, 8PSK E 16QAM. Este módulo recebe os símbolos que já foram mapeados para a constelação, e a partir dos mesmos realiza a correta filtragem do sinal a partir do filtro SRRC.

A figura 32 apresenta o diagrama utilizado no módulo de modulação. Os símbolos complexos são recebidos como uma memória de dados, em seguida é realizada a sobre amostragem do sinal a partir de um fator L o qual este relacionado ao filtro SRRC. Após a sobre amostragem do sinal, utilizando uma convolução linear, os símbolos são convolvidos com os coeficientes do filtro SRRC presente em uma memória de dados.

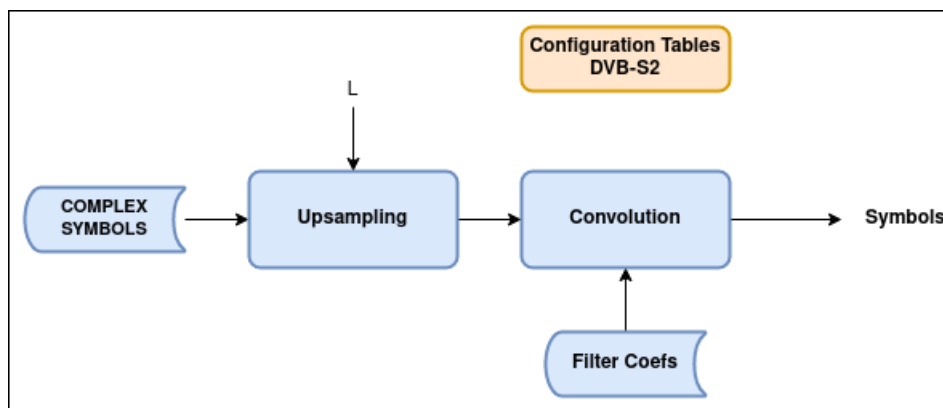


Figura 32 – Diagrama de blocos do módulo *Modulation* do codificador RCS2

Após a convolução, a última etapa do sistema consiste em multiplicar o sinal por um módulo oscilador numericamente controlado, de forma definir a frequência de oscilação do sistema.

3.1.2 Spreading

Este módulo é dividido em *Spread* e *un-spread*, as informações sobre o tipo da codificação a ser utilizada, estão presentes nas tabelas *Format Data Blocks*.

A figura 33 apresenta o diagrama de blocos deste módulo. Os blocos *INS.SYMBS* e *INS.CHIPS* são responsáveis por adicionar símbolos de preâmbulo, piloto e pós-âmbulo no *burst* recebido. O sistema recebe um conjunto de símbolos a partir das informações presentes nas tabelas de configuração DVB-S2, de forma que a partir do sinal *SPREAD* é possível verificar se o tipo de modulação é *TC-LM* ou *SS-LM*.

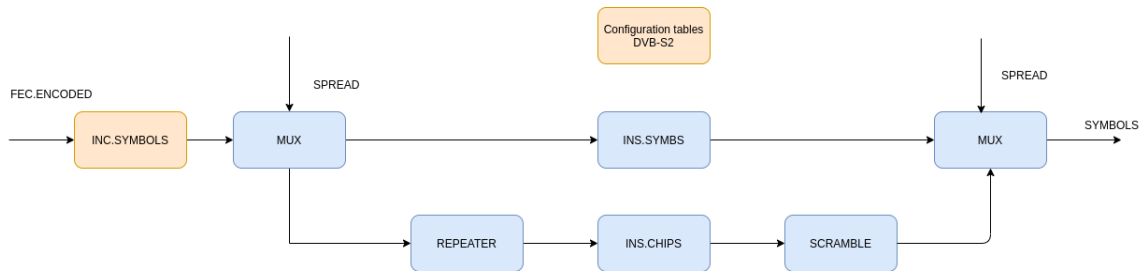


Figura 33 – Diagrama de blocos do módulo *spreading* do codificador RCS2

A figura 34 apresenta o formato onde os símbolos devem ser inseridos no sistema, onde as seções em verde representam em tamanhos fixos os blocos de *payload*. As informações para a correta organização desta estrutura estão presentes nas tabelas de configuração, as quais podem ser vistas no anexo A da norma (ETSI, 2014-04). Os símbolos de *pre-amble(pre)*, *pos-amble(pos)*, *pilot(P)* e *payload(em verde)* devem ser inseridos de acordo com a figura 34 formando desta forma o burst.



Figura 34 – Inserção de símbolos de preâmbulo e pós-âmbulo do bloco *INS.SYMBS* do codificador RCS2. Retirado de (ETSI, 2014-04)

As tabelas apresentam informações adicionais, de forma que os símbolos a serem inseridos podem ser customizados ou recebidos diretamente de uma tabela de configurações presente na memória do sistema.

O módulo *REPEATER* é utilizado para repetir os símbolos um número de vezes igual ao fator de espalhamento, o qual esta presente na tabela *Format Data Block for Spread-Spectrum LM Burst* presente em (ETSI, 2014-04).

O bloco de *SCRAMBLE* é construído a partir da multiplicação símbolo a símbolo por uma sequência de valores presentes na memória do sistema. A cada novo símbolo que entra no sistema, o índice de memória da tabela apresentada na figura 35 é incrementada de forma cíclica até que todos os símbolos sejam multiplicados. A equação a seguir apresenta a operação matemática do sistema, de tal forma que $Z[n]$ consiste no valor obtido

da tabela contendo as seqüências de espalhamento.

$$Y[n] = X[n]Z[n] \tag{3.1}$$

| | | | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 1 | 1 | 1 | 1 | -1 | 1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 | -1 | -1 | 1 | -1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | -1 | 1 | 1 | 1 | -1 | -1 | -1 | 1 | 1 | 1 | -1 | 1 |
| 1 | -1 | 1 | -1 | -1 | -1 | 1 | 1 | 1 | 1 | 1 | 1 | -1 | -1 | 1 | 1 | -1 | 1 |
| 1 | -1 | 1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 | -1 | -1 | 1 | 1 | 1 | 1 |
| -1 | 1 | -1 | 1 | 1 | -1 | 1 | -1 | -1 | 1 | -1 | 1 | -1 | -1 | 1 | 1 | -1 | 1 |
| -1 | -1 | 1 | 1 | 1 | 1 | -1 | -1 | -1 | -1 | 1 | 1 | -1 | -1 | 1 | -1 | -1 | -1 |
| -1 | 1 | -1 | 1 | -1 | -1 | 1 | -1 | -1 | 1 | 1 | 1 | -1 | 1 | 1 | 1 | 1 | -1 |
| -1 | 1 | -1 | -1 | 1 | 1 | 1 | -1 | -1 | -1 | -1 | 1 | 1 | -1 | -1 | -1 | 1 | 1 |
| -1 | -1 | 1 | 1 | 1 | -1 | 1 | -1 | -1 | 1 | -1 | 1 | 1 | -1 | 1 | 1 | 1 | -1 |
| 1 | -1 | -1 | 1 | 1 | -1 | -1 | -1 | -1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | -1 |
| 1 | 1 | -1 | -1 | 1 | 1 | -1 | -1 | 1 | -1 | 1 | -1 | 1 | 1 | 1 | -1 | -1 | 1 |
| 1 | 1 | 1 | 1 | -1 | -1 | -1 | 1 | -1 | 1 | -1 | 1 | -1 | 1 | -1 | -1 | -1 | -1 |
| -1 | 1 | 1 | -1 | 1 | -1 | 1 | 1 | 1 | 1 | -1 | 1 | 1 | -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | 1 | | | | | | | | | | | | | | |

Figura 35 – sequência padrão de espalhamento do codificador RCS2 retirado de (ETSI, 2014-04)

O protocolo DVB-RCS2 sugere uma equação para a geração destes símbolos, os quais pode ser observado na figura 15.

3.2 Multi-frequency time-division multiple access

3.2.1 Tabelas e descritores

Após a leitura do protocolo DVB-RCS2 é observado que o fluxo de implementação do *Multi-frequency time-division multiple access* deve ser iniciado a partir da construção das tabelas e descritores do sistema. A figura 36 apresenta os principais campos que devem ser consultados para a construção do sistema, os quais referir-se a outras tabelas na norma.

| | | | | | | | | | | | |
|----------------|------------------------|------|-----|-----|-------|------|-------|-----|-----|-------|--------------|
| NCR | SCT | FCT2 | BCT | SPT | TMST2 | MMT2 | TIM-B | FAT | CMT | TBTP2 | TIM-U |
| | Broadcast Table Format | | | | | | | | | | |
| Unlabelled GSE | | | | | | | | | | | 6B Label GSE |
| DVB-S2 | | | | | | | | | | | |

Figura 36 – Principais tabelas utilizadas no MF-TDMA (ETSI, 2014-04)

Cada uma destas tabelas apresenta um conjunto de informações necessárias a implementação do sistema.

- SCT: Apresenta a composição básica do *super frame*.
- FCT2: Apresenta todos os tipos de *frames* do sistema.
- BCT: Apresenta o tipo de transmissão e codificação dado um tipo de frame.
- TBTP2: Apresenta informações relativas à organização do super frame e ao tipo de acesso.
- TIM-U: Apresenta informações de controle do sistema
- TIM-B: Apresenta informações de correção de mensagem a serem aplicadas a um terminal.

A partir do uso das tabelas da figura 36 é possível construir a estrutura de controle do MF-TDMA e de suas funções internas descrita nas seguintes seções.

3.2.2 Máquina de estado principal

A implementação da máquina de estado principal foi feita utilizando laços condicionais com a chamada de funções que representam cada um dos estados. O controle de transição de estados é feito a partir da verificação da resposta de cada estado, bem como da constante verificação da tabela TIM-U presente na figura 36. A tabela TIM-U apresenta os sinais utilizados para transição de estados, os quais podem ser visto na figura 37.

- Off/Standby: Este é o estado normal imediatamente após a inicialização, bem como um estado padrão para o qual o terminal retorna em algumas situações após a perda de sincronização ou ao ser desconectado. O link direto deve ser mantido operacional neste estado. Ao entrar o estado Off/Standby, o terminal cessará imediatamente a transmissão. Pode manter identificadores dinâmicos se especificamente autorizado a fazê-lo conforme indicado para a tarefa. O terminal não deve transmitir enquanto estiver no estado Off/Standby.
- Hold/Standby: Ao entrar no estado Hold/Standby, o terminal interromperá imediatamente a transmissão. Um terminal no estado Hold/Standby deve permanecer lá após os eventos de reinicialização até que o NCC libere a(s) condição(ões) que mantêm o terminal no estado Hold/Standby. O terminal não transmitirá enquanto estiver no estado Hold/Standby.
- Ready for Logon: O terminal entra neste estado quando o link direto foi adquirido com sucesso e os dados de configuração necessários para emitir o logon estão atualizados. Gatilhos externos podem incluir, por exemplo, chegada de dados na interface

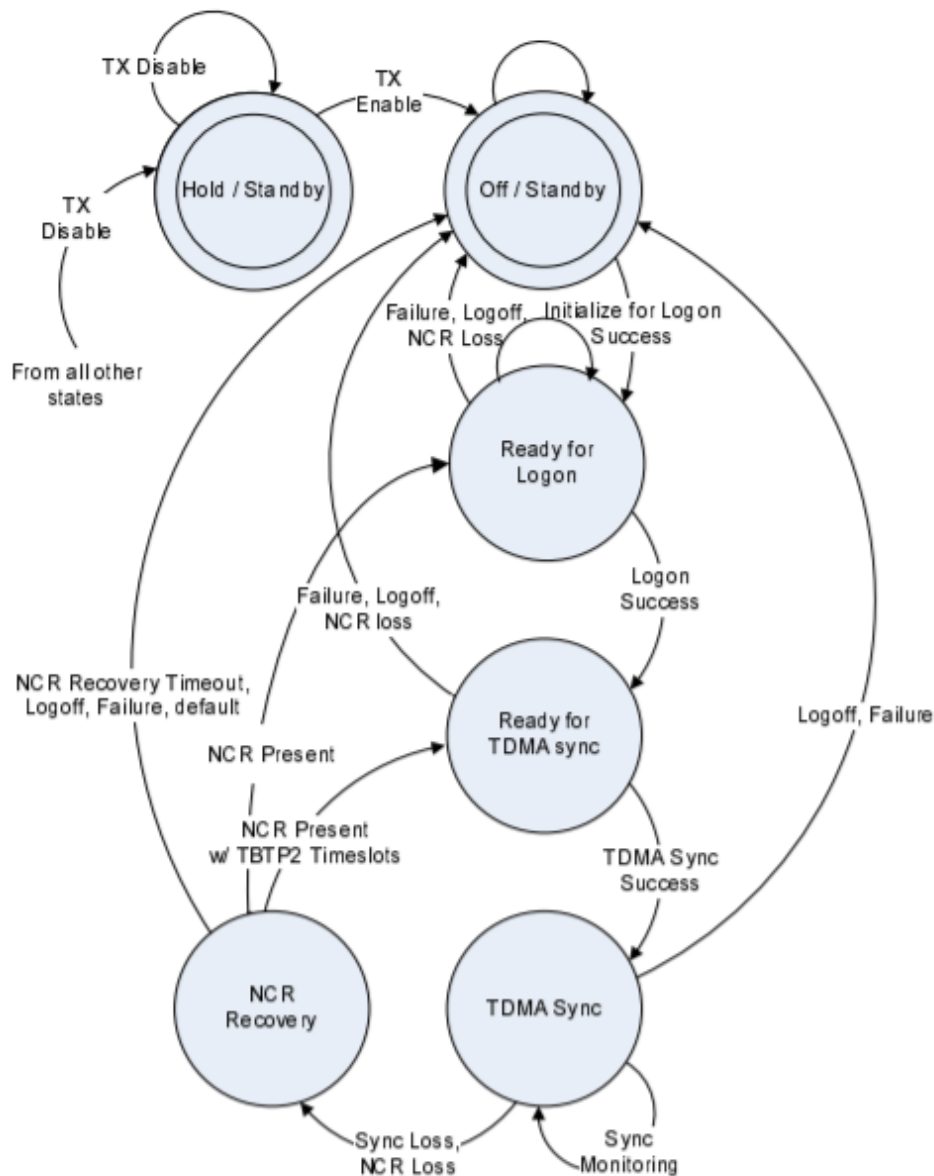


Figura 37 – MF-TDMA: FSM principal (ETSI, 2014-04)

terrestre ou recepção de uma mensagem de “despertar” no TIM-U. A transmissão de rajadas de logon é permitida quando o terminal está nesse estado.

- Ready for TDMA Sync: O terminal está neste estado quando o procedimento de logon da camada inferior foi concluído com sucesso, mas a sincronização TDMA ainda não foi alcançada. Portanto o terminal deve transitar para outro estado. A transmissão de rajadas de controle é permitida quando o terminal está neste estado.
- TDMA Sync: Este é o estado operacional normal para o terminal. Este é um estado absorvente qual faz com que o terminal permanecerá lá até que eventos externos ou perda de sincronização ditem a transição para outro estado. A transmissão de rajadas de controle é permitida quando o RCST está neste estado. A transmissão

de rajadas de tráfego e rajadas de tráfego/controlado podem ser permitidas ou podem ser bloqueado dinamicamente.

- NCR Recovery: O RCST entra neste estado quando há perda de sincronização de TDMA ou perda de NCR quando em TDMA Sincronizar. Este é um estado não absorvente; o RCST transitará de forma autônoma para outro estado. O RCST não transmitir enquanto estiver no estado NCR Recovery.

3.2.3 Algoritmo de *logon*

A figura 38 apresenta o algoritmo de *logon* do sistema. O seu funcionamento é baseado nas respostas obtidas das tabelas TIM-U e TIM-B e das funções internas do sistema. Este estado realiza uma tentativa de *logon* na rede, e possui 2 formas básicas, as quais são por *Random Access* e *Dedicated Access*. Cada um desses tipos de acesso possui uma função a ser implementada, o qual retorna a frequência em *Hz* e o tempo em *NCR ticks* para uma função de transmissão. A tentativa de *logon* é efetuada após a seleção do *timeslot* para *logon* no estado *select DA logon burst* ou *select RA logon burst*. O *Burst* contendo as informações de *logon* é enviado á HUB utilizando o bloco *Transmit Logon burst*, de forma que o sistema ira aguardar o recebimento das informações em suas tabelas e descritóres.

A função de transmissão recebe como parâmetros dados de tempo e frequência e realiza a codificação e transmissão do sinal. Por fim, é feita uma verificação da resposta obtida da HUB para verificar a qualidade do sinal transmitido, determinando a necessidade da aplicação de correções no sinal.

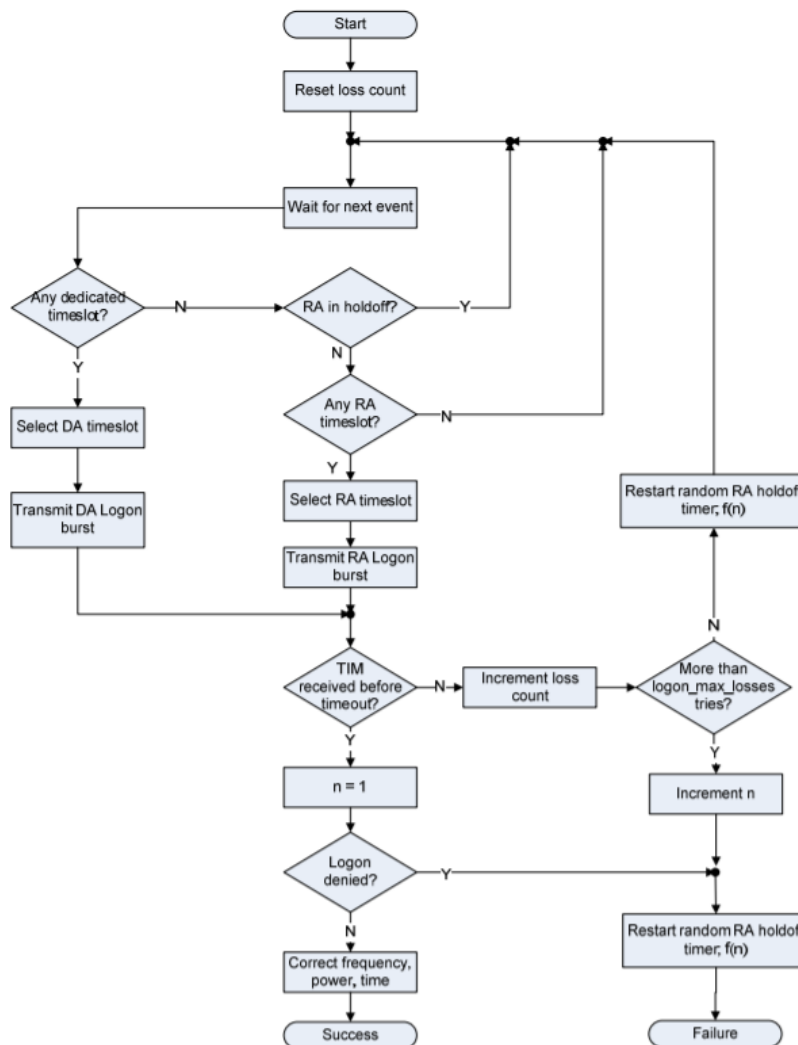


Figura 38 – Ready for Logon(ETSI, 2014-04)

As funções para seleção de *timeslots* são divididas em acesso dedicado e acesso aleatório. A seguir será apresentado o diagrama de blocos proposto para obtenção de um *timeslot* de acesso aleatório.

A primeira etapa, apresentada na figura 39, consiste em consultar a tabela FCT2 e selecionar dentre os *frames* para logon, um *frame* que contenha *timeslots* para acesso aleatório. Em seguida, é selecionado de forma aleatória um *timeslot*. A partir do *timeslot* é feita uma soma das durações (em contagens de NCR) do *super frame*, *frame* e *timeslot* para obter no tempo e na frequência a correta posição do *timeslot*.

O procedimento para obter o *timeslot* para controle é feita de maneira direta a partir do descritor *Control Assign Descriptor*, o qual indica a correta frequência e posição do *timeslot* no tempo e na frequência.

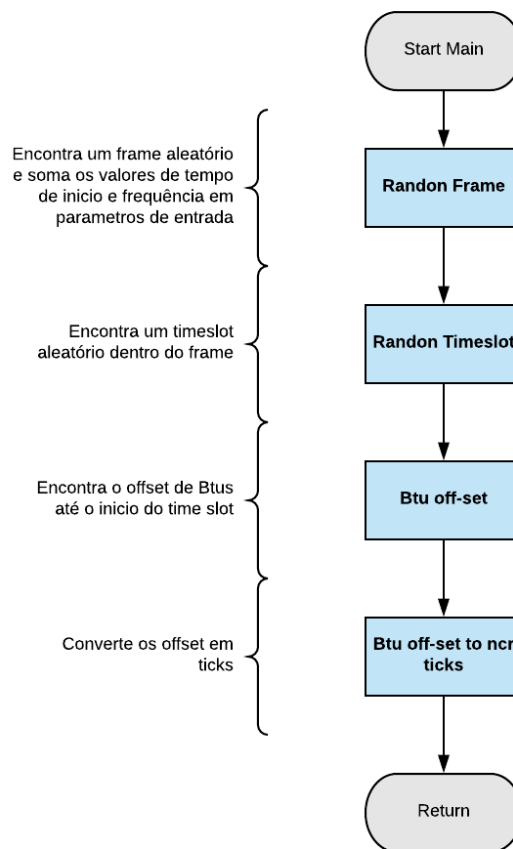


Figura 39 – Acesso randômico

3.2.4 Algoritmo de sincronização

As figuras 40 e 41 apresentam os algoritmos utilizados para a construção dos estados *Ready for TDMA sync* e *TDMA Sync*, respectivamente. Ambos os estados são utilizadas para realizar a transmissão de um *burst* de controle. Ambos os estados de sincronização apresentam funções idênticas, entretanto diferem-se na forma de retorno e verificação dos sinais de controle obtidos pelas tabelas e descritores.

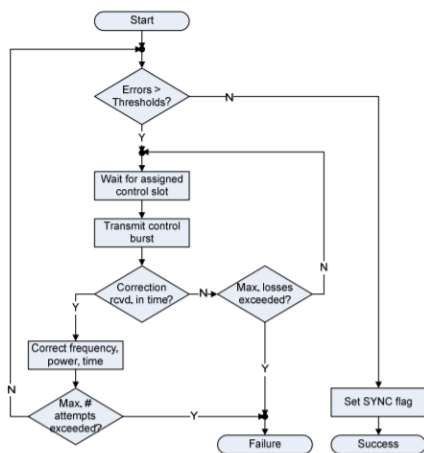


Figura 40 – Ready for TDMA sync(ETSI, 2014-04)

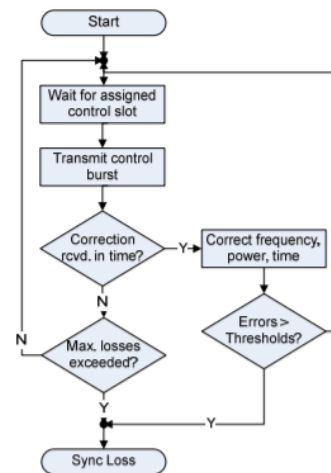


Figura 41 – TDMA Sync(ETSI, 2014-04)

3.2.5 Sincronismo de relógio - NCR

A recuperação do sincronismo entre o clock da HUB com o clock do terminal foi desenvolvido com base no trabalho presente em (LISS, 2012). O trabalho apresenta a descrição dos blocos fundamentais no circuito de recuperação de relógio. Desta forma, foi possível implementar um circuito ADPLL com base na descrição dos blocos apresentados no trabalho. O diagrama da figura 42 apresenta a arquitetura do circuito ADPLL a ser desenvolvido neste trabalho.

3.2.5.1 Recepção dos TS-packet

A primeira etapa do sistema consiste em receber um pacote que contenha um valor de PCR e em seguida salvar uma marca de tempo no sistema. Esta marca de tempo é feita pelo sinal *load* presente na figura 43.

Após isto é verificada a diferença entre a marca de tempo local, e a recebida (PCR). O erro entre os dois sistemas determina o valor do contador local, pois em caso deste erro ser muito grande, é necessário carregar o valor recebido de PCR no contador local. A seguir será apresentado o algoritmo que controla o recebimento dos pacotes e a obtenção do erro.

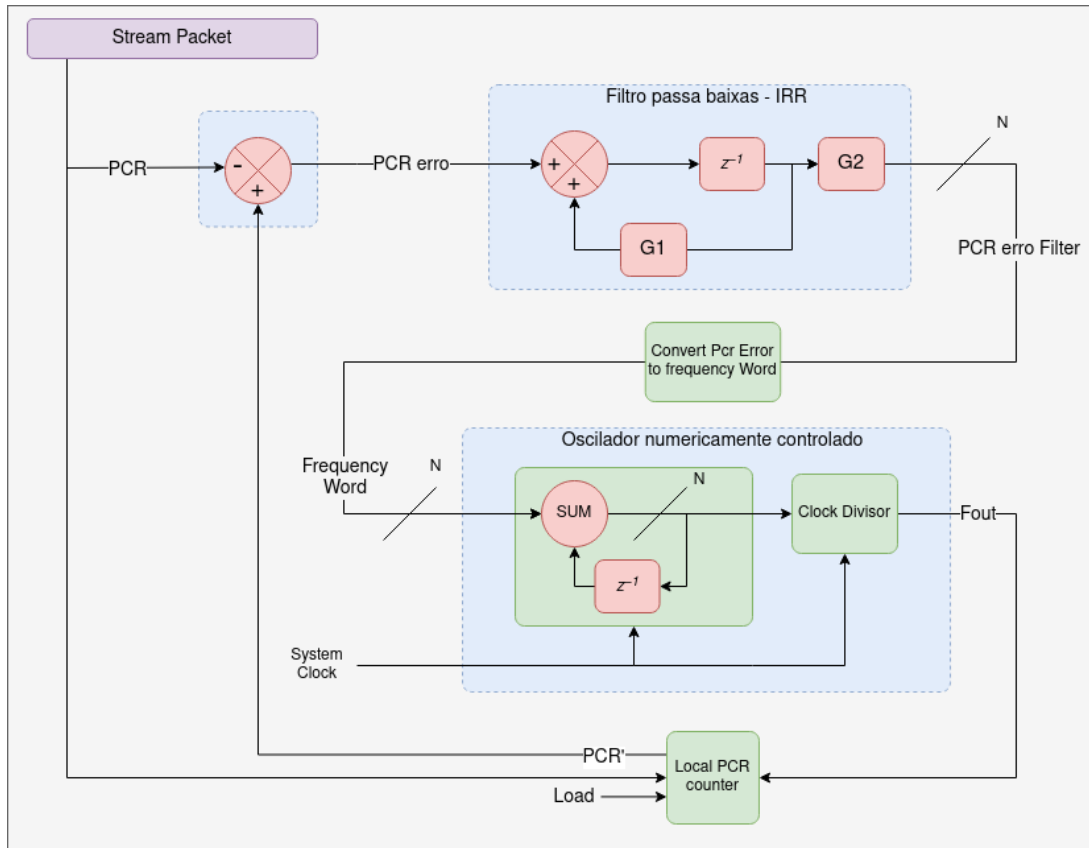


Figura 42 – Arquitetura geral do sistema (ETSI, 2014-04)

3.2.5.2 Controlador

O controlador consiste de um sistema que recebe um valor de erro como entrada e realiza a filtragem, para eliminação de componentes de alta frequência, como jitter. O sistema é baseado em um controlador do tipo P. O sistema consiste de um filtro passa baixas que possui o seguinte diagrama de blocos.

A implementação digital deste filtro consiste na utilização de um filtro IRR, sendo este um filtro de média cuja equação do filtro dividida em duas partes. A primeira definida como $Y[n]$ consiste na saída do módulo Z^{-1} , enquanto a saída do filtro é definida como $S[n]$ e consiste na saída do módulo $G2$.

$$y[n] = x[n - 1] + y[n]G1 \quad (3.2)$$

$$S[n] = y[n]G2 \quad (3.3)$$

As simulações do filtro feitas em (LISS, 2012) indicam que o valor dos coeficientes do filtro devem ser *Gain* : $G1 = 0.99$ e *Scaling* : $G2 = 2^{-17}$. O resultado da saída deste

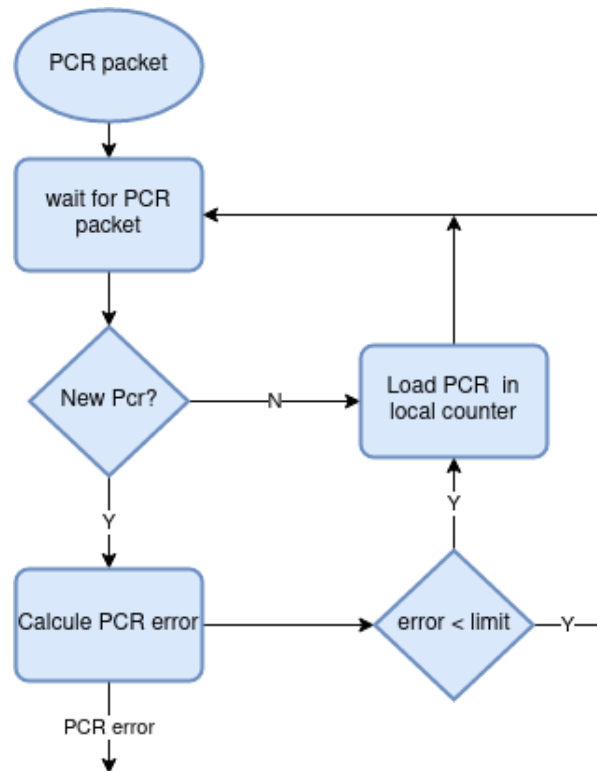


Figura 43 – Algoritmo de recepção dos ts packet

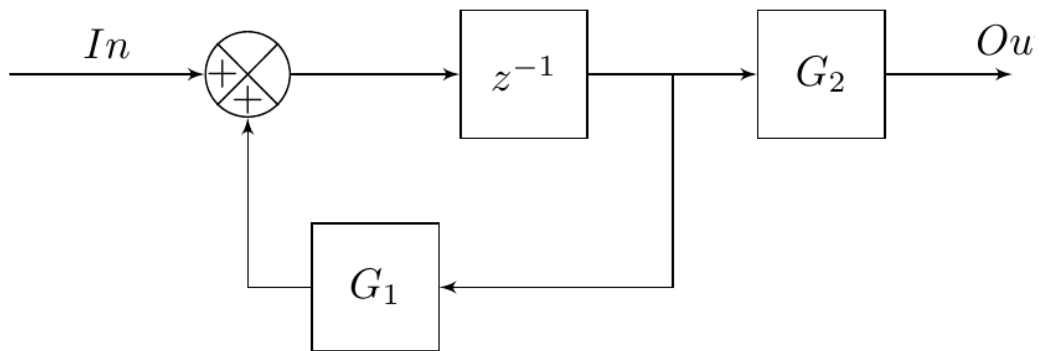


Figura 44 – filtro passa baixas

filtro é direcionado ao bloco que converte esse erro em um desvio de frequência, o qual controla um oscilador numericamente controlado.

3.2.5.3 System Time Clock (STC)

O System Time Clock (STC) consiste em um clock de 27 MHz o qual deve permitir que a sua frequência seja regulada, com o intuito de permitir a correção de relógio. O sistema é composto por um oscilador numericamente controlado, o qual utiliza a saída do controlador P, para realizar a correção de frequência do sinal. Desta forma, com o controle do clock de 27 MHz, o sistema é capaz de sincronizar os NCRs do receptor com o do transmissor.

3.2.5.4 Contador Local

Este módulo é construído como um contador de 42 bits que recebe as entradas *load*, *clk*, *save*, PCR_{in} e tem como saída PCR_{OUT} . O *load* é utilizado para carregar o valor de PCR_{in} no contador, e o sinal *save* é utilizado para salvar o valor do contador no momento em que o sistema receber um novo valor de PCR. O módulo deve retornar a zero sempre que atingir $(2^{33} - 1)300 + 299$.

3.3 Ferramentas

Neste trabalho, serão utilizadas as seguintes ferramentas para o desenvolvimento do MF-TDMA e do sincronizador.

- Python (numpy, matplotlib, scipy, serial,): A linguagem python é utilizada para apresentar de forma gráfica a transição de estados do MF-TDMA. Também foi utilizada para realizar a sincronia de tempo entre uma *esp32* e o relógio do MF-TDMA.
- C: Linguagem de programação de propósito geral. É utilizada para codificar os módulos implementados em ARM.
- VHDL: Linguagem de descrição de hardware. É utilizada para codificar os módulos em hardware do sistema.
- VIVADO: Ambiente para a descrição em hardware dos componentes, síntese, implementação física dos circuitos. Foi feita o encapsulamento de um IP utilizando a Interface AXI4-Lite integrando a arquitetura desenvolvida em hardware com a aplicação em software.
- GNURadio: Software para o estudo de caso do MF-TDMA para o envio e recepção de sinais de RF utilizando a interface com o SDR Adalm pluto.
- Adalm Pluto: A placa SDR Adalm Pluto da empresa Analog Devices para enviar sinais RF em uma curta distância para o transceiver *FMCOMMS3* afim de enviar e receber sinais.

- AD9361: O AD9361 é um Agile Transceiver™ de rádio frequência (RF) altamente integrado e de alto desempenho projetado para uso em aplicações de estação base 3G e 4G. Sua programação e capacidade de banda larga o tornam ideal para uma ampla gama de aplicações de transceptores. O receptor LO AD9361 opera de 70 MHz a 6,0 GHz e o transmissor LO opera na faixa de 47 MHz a 6,0 GHz, abrangendo a maioria das bandas licenciadas e não licenciadas. Larguras de banda de canal de menos de 200 kHz a 56 MHz são suportadas.
- ESP32: Microcontrolador o qual foi utilizado para apresentar o correto funcionamento do *MF-TDMA* no tempo.
- ZedBoard: O kit de desenvolvimento que foi utilizado é o kit ZedBoard Zynq-7000 Development Board, mostrado na figura 45. O kit conta com um SoC XC7Z020-CLG484 cujas principais características são: processador ARM Cortex-A9 Dual Core 667MHz, 512 MB DDR3, 256 Mb Quad-SPI Flash, 4 GB SD card, USB 2.0, USB-UART, 10/100/1000 Ethernet, 85K logic cells, 13300 logic slices, cada um com 4 LUTs de 6 entradas 8 flip-flops ([DIGILENT](#),).

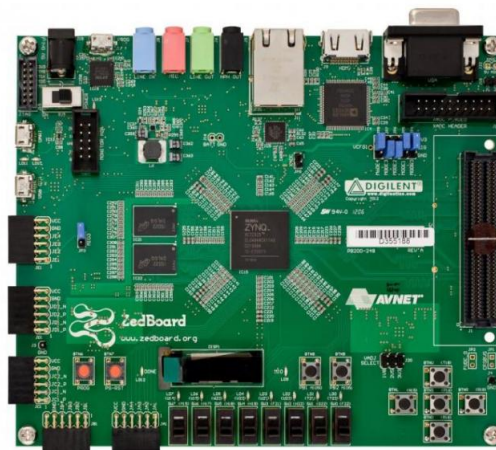


Figura 45 – Kit de desenvolvimento Zybo Zynq-7000 ARM/FPGA SoC Trainer Board ([DIGILENT](#),)

4 Proposta de validação e implementação

4.1 *Modulation and Spreading*

Os blocos do codificador RCS2 implementados em ARM podem ser verificados a partir do funcionamento geral do sistema, tendo como referência o modelo em Python. Ademais, os blocos individuais de modulação apresentam modelos em Octave, os quais apresentam as formas de onda geradas na saída do modulador. A verificação deste módulo é feita a partir da análise da constelação gerada para cada um dos tipos utilizados. Os módulos do codificador RCS2 não necessitaram de uma implementação em hardware, pois a implementação em software conseguiu satisfazer os critérios para o tempo de codificação do sistema.

4.2 *Multi-frequency time-division multiple access*

A validação do MF-TDMA foi feita de duas formas distintas. A primeira a partir da leitura e escrita das tabelas e descritores validando a correta transição dos estados do sistema. Foram utilizados valores de teste, os quais devem ser inseridos nas tabelas e descritores do sistema. Com auxílio do software LabView, foram verificadas as transições de estado bem como a reposita de algumas funções internas do sistema. Os valores utilizados para o teste, bem como para o estado atual e o próximo estado, estão apresentados na tabela 2.

A validação das funções de obtenção dos *timeslots* para acesso aleatório e dedicado foi feita a partir do ambiente de simulação criado no LabView, o qual apresentará um campo indicando a *frequência* e o tempo obtido de um determinado *timeslot*. Os valores de teste criados para as funções de obtenção dos *timeslots* utilizaram como base um *super frame* contendo um *frame* com 16 *timeslots*. O super frame de teste possui as características apresentadas na tabela 3.

Tabela 2 – Condições de teste do MF-TDMA

| Condições | Estado Atual | Proximo Estado |
|---------------------|-----------------|-----------------|
| Tx_Disable = 0 | Hold-Standby | Hold-Standby |
| Tx_Disable = 1 | Hold-Standby | Off-Standby |
| Tx_Disable = 0 | Off-Standby | Hold-Standby |
| InitLogon = 0 | Off-Standby | Off-Standby |
| InitLogon = 1 | Off-Standby | Ready-for-logon |
| Failure = 1 | Ready-for-logon | Off-Standby |
| Logoff = 1 | Ready-for-logon | Off-Standby |
| NCR Loss = 1 | Ready-for-logon | Off-Standby |
| LogonSucess = 1 | Ready-for-logon | ReadyTdmaSync |
| Failure = 1 | ReadyTdmaSync | Off-Standby |
| Logoff = 1 | ReadyTdmaSync | Off-Standby |
| NCR Loss = 1 | ReadyTdmaSync | Off-Standby |
| TdmaSyncSuccess = 1 | ReadyTdmaSync | TdmaSync |
| Failure = 1 | TdmaSync | Off-Standby |
| Logoff = 1 | TdmaSync | Off-Standby |
| NCR Loss = 1 | TdmaSync | Ncr-Recovery |
| Sync Loss = 1 | TdmaSync | Ncr-Recovery |
| NCR Present = 1 | Ncr-Recovery | Ready-for-logon |
| NcRr_TBPT2 = 1 | Ncr-Recovery | ReadyTdmaSync |
| NcRecovTimeOut = 1 | Ncr-Recovery | Off-Standby |
| Logoff = 1 | Ncr-Recovery | Off-Standby |
| Failure = 1 | Ncr-Recovery | Off-Standby |
| Default = 1 | Ncr-Recovery | Off-Standby |

Tabela 3 – Super frame de teste

| | |
|------------------------------|---------|
| Frequência Central | 2.0 Ghz |
| Duração | 1 s |
| Numero de timeslots | 10 |
| Timeslots de controle | 8 |
| Timeslots de logon | 1 |
| Timeslots de acesso dedicado | 1 |

A segunda forma de validação foi feita emulando uma HUB utilizando o software GNU Radio em conjunto com a SDR Adalm Pluto e o transceiver FCOMMS3. O esquemático da Figura 46 foi utilizado como simulação de uma HUB.

O bloco *Signal Source* é responsável por criar um sinal senoidal o qual foi transmitido pelo bloco PlutoSDR Sink. A partir da alteração da frequência da senoide transmitida, o MF-TDMA recebe o sinal e faz uma FFT, obtendo assim a sua frequência. Esta frequência foi utilizada em uma função do MF-TDMA para tomar certas ações, como alteração dos valores de tabelas, descritores ou reconfigurar os *timeslot* para transmissão. Os blocos de configuração do MF-TDMA podem ser vistos nas figuras 41, 40 e 38 como

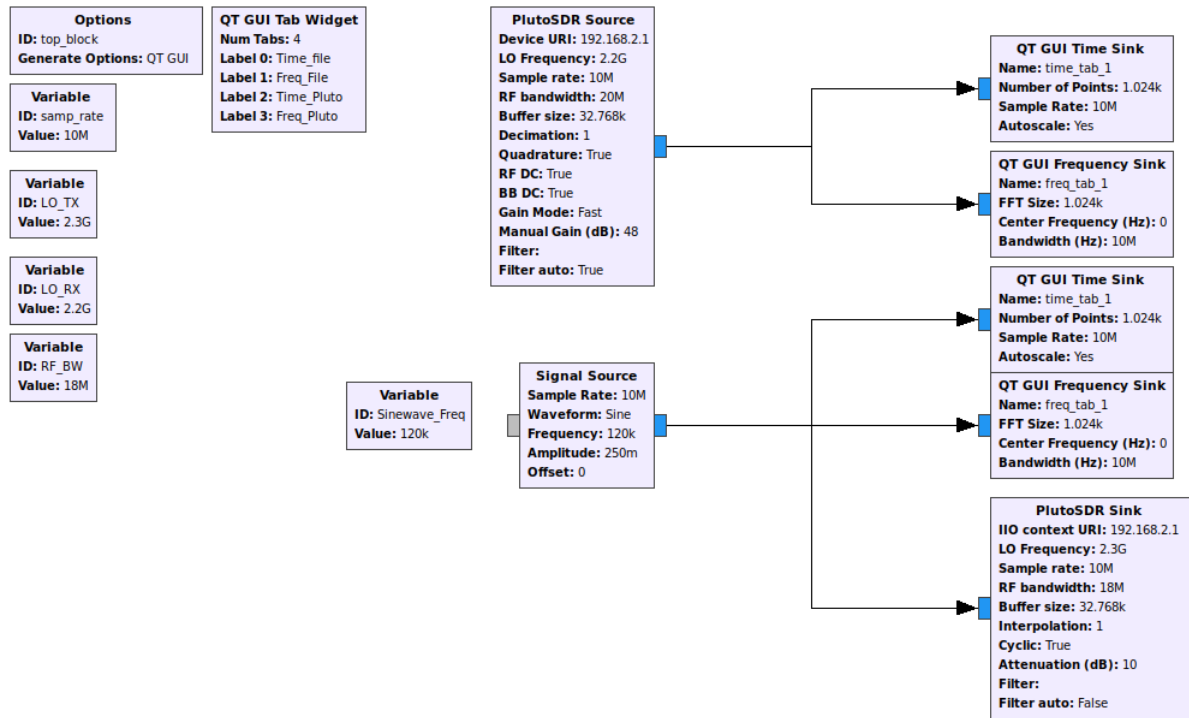


Figura 46 – Simulação de uma HUB utilizando o GNU radio.

o nome de *Wait for assigned control slot*. Para o caso dos timeslot selecionados durante a etapa de logon, o sistema identifica em suas tabelas, quais timeslots podem ser utilizados para logon e os utiliza a partir de uma seleção aleatória dos slots disponíveis.

Tabela 4 – Condições de teste do MF-TDMA no GNU radio

| | |
|--------------------|-----------------|
| Sinewave Frequency | |
| 40 Khz | Logoff = 1 |
| 60 Khz | Tx_Disable = 0 |
| 80 Khz | Tx_Disable = 1 |
| 100 Khz | LogonSucess = 1 |
| 120 Khz | InitLogon = 1 |
| 140 Khz | InitLogon = 0 |
| 160 Khz | Sync Loss = 1 |
| 180 Khz | Sync Loss = 0 |
| 200 Khz | Failure = 0 |

A tabela 4 apresenta os valores de Frequência da senoide enviada para que o MF-TDMA altere a suas tabelas e descritores resultando nas transições de estado de acordo com a tabela 2.

A reconfiguração dos *Timeslot* é feita de forma semelhante. O descritor *Control Assign Descriptor* é reconfigurado de acordo com o valor recebido de frequência em todos os estados responsáveis por solicitar esta informação da HUB, conforme a tabela 5.

Tabela 5 – Reconfiguração dos *timeslot*

| Sinewave Frequency | |
|--------------------|-------------|
| 120 Khz | Timeslot-3 |
| 140 Khz | Timeslot-4 |
| 160 Khz | Timeslot-5 |
| 180 Khz | Timeslot-6 |
| 200 Khz | Timeslot-7 |
| 220 Khz | Timeslot-8 |
| 240 Khz | Timeslot-9 |
| 260 Khz | Timeslot-10 |

Tabela 6 – Super Frame

| frame | Duração | Frequência |
|-------|---------|------------|
| 1 | 1000 ms | 2.2Ghz |

Tabela 7 – Frame

| Timeslot | Início do timeslot | Frequência | Tipo |
|----------|--------------------|------------|----------|
| 1 | 0 ms | 2.2Ghz | Logon |
| 2 | 200 ms | 2.2Ghz | Logon |
| 3 | 400 ms | 2.2Ghz | Controle |
| 4 | 600 ms | 2.2Ghz | Controle |
| 5 | 800 ms | 2.2Ghz | Controle |
| 6 | 0 ms | 2.325Ghz | Controle |
| 7 | 200 ms | 2.325Ghz | Controle |
| 8 | 400 ms | 2.325Ghz | Controle |
| 9 | 6000 ms | 2.325Ghz | Controle |
| 10 | 8000 ms | 2.325Ghz | Controle |

A construção e configuração dos *timeslot* é feita com base na construção de um super frame contendo um único *frame* com duração de 1 segundo e frequência central de 2.2 Ghz conforme as tabelas 6 e 7.

4.2.1 Controle do tempo e validação

O MF-TDMA deve ser capaz de realizar a transmissão de um sinal, na janela de tempo permitida para o *timeslot* em uso. Para tal, foi desenvolvido um módulo em hardware para realizar uma interrupção, indicando para o sistema ARM, que a transmissão deve ser iniciada ou finalizada. O controle do início e fim das transmissões é feita a partir das funções do módulo AD9361. A configuração deste módulo pode ser feita a partir da aplicação em ARM, através do endereçamento dos registradores utilizando o barramento AXI-Lite, permitindo a configuração do tamanho máximo da soma de todos os super frames do sistema. Desta forma, o contador NCR, irá sempre voltar a zero quando atingir o valor máximo de contagem, definindo assim uma janela de tempo, para que o MF-TDMA

possa realizar as transmissões. A arquitetura desse bloco é apresentada a seguir.

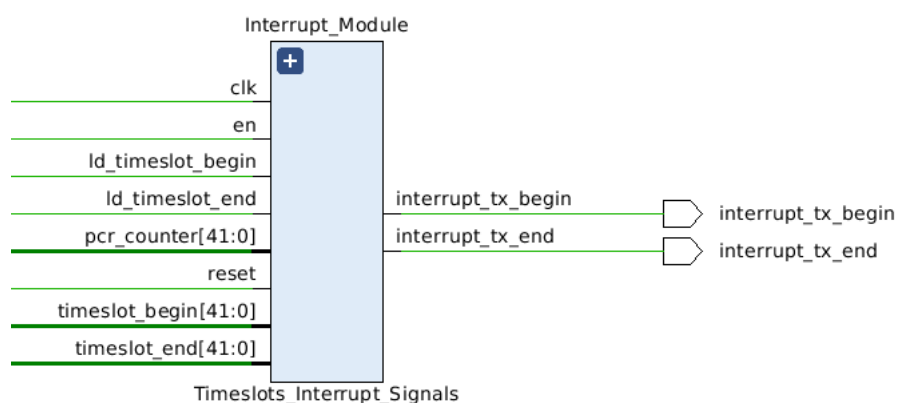


Figura 47 – Bloco de controle das transmissões do MF-TDMA

A validação da correta transmissão dos sinais no tempo é feita utilizando o microcontrolador ESP32 em conjunto com Python de acordo com o diagrama de blocos da figura 48. Foi criado um programa na ESP32 contendo um contador de 0 a 5 segundos. Este contador acende um LED e escreve na porta serial que a janela de transmissão do *timeslot* configurado está ativa. A sincronização do relógio da ESP32 com o do MF-TDMA é feita via serial UART. Quando o MF-TDMA é iniciado, o sistema zera sua referência de tempo e escreve na porta serial uma palavra-chave, informando a ESP32 que deve setar a sua referência de tempo para zero. O bloco em verde apresenta o MF-TDMA implementado em linguagem C em conjunto com os módulos em hardware implementados em VHDL. O bloco em bege

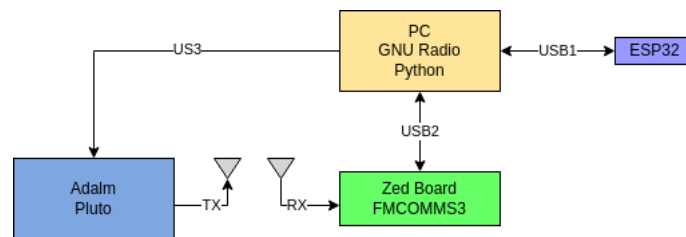


Figura 48 – Arquitetura para a transmissão no tempo

Com uma referência de tempo externa ao sistema, a verificação da janela de tempo pode ser efetuada visualmente. Desta forma, para realizar a validação, basta observar o LED da ESP32 ou a porta serial junto ao bloco de visualização do GNU Radio verificando a aparição de um sinal durante a janela de transmissão do sistema.

4.2.2 Verificação na frequência de transmissão

A validação da frequência do MF-TDMA foi feita utilizando o bloco PlutoSDR Source do GNU Radio. A Tabela 4 apresenta uma variável de controle da frequência deste módulo. A mesma deve ser configurada conforme a frequência do *timeslot*, o qual pode ser observado na Tabela 7. Desta forma, foi observado que em certos intervalos de tempo definidos, o sistema recebe o sinal DVB-RCS2 enviado pelo MF-TDMA.

4.3 Sincronismo de relógio - NCR

O controle da transmissão e sincronia de relógio é feita por um IP Block criado a partir do Vivado, a FSM do sistema é apresentada na figura 49.

Na figura 49 o estado de *TRANSMIT* é responsável por iniciar a transmissão, ativando o módulo da Figura 47. O estado *SINCRONIZE* é responsável por realizar a sincronização com um contador de referência. A arquitetura desse IP é apresentada na Figura 50.

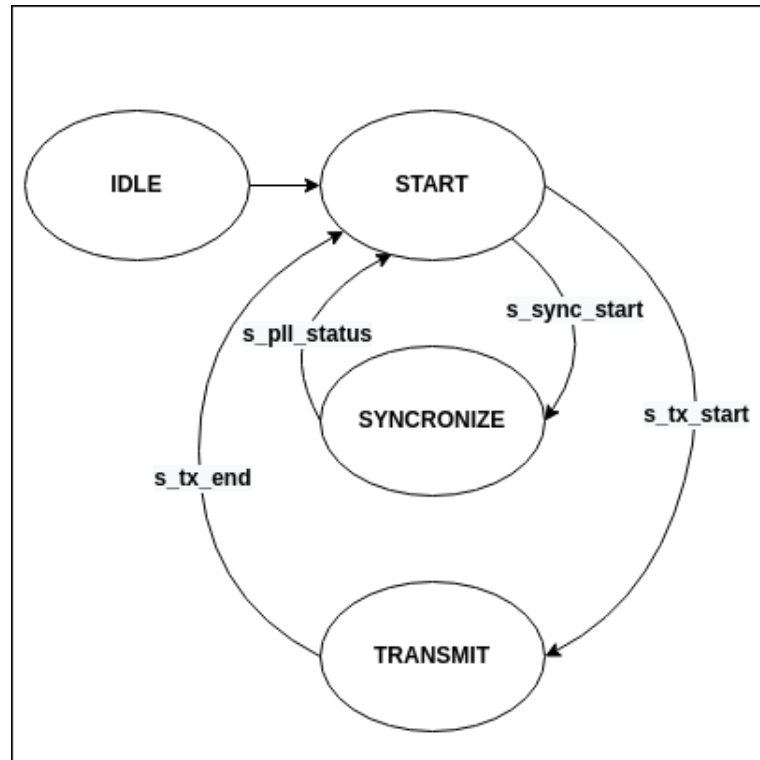


Figura 49 – TIME SYNC - FSM

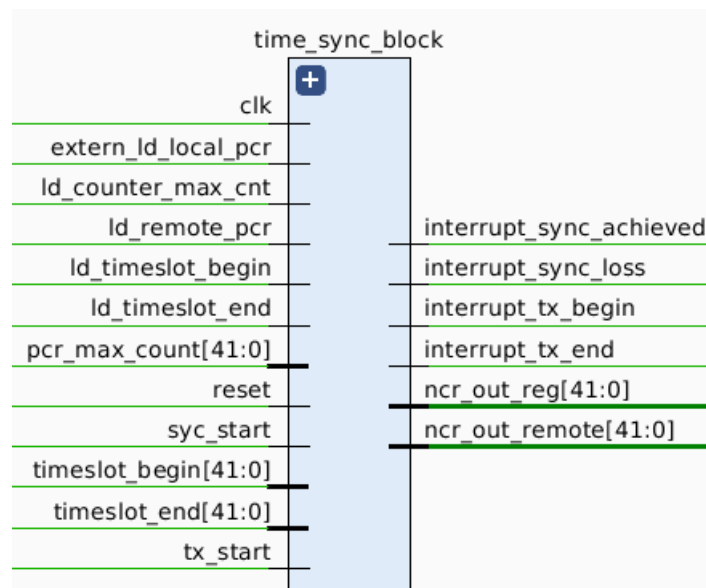


Figura 50 – TIME SYNC IP

O sistema da figura 50 possibilita o controle da sua FSM interna partir dos sinais `sync_start` e `tx_start`. As entradas do módulo são controladas via barramento AXI-Lite, de forma que é utilizado 1 registrador de 32 bits para controlar os sinais de controle do IP e 2 registradores de 32 bits para configurar os registradores de 42 bits do bloco.

O sistema possui 2 saídas de interrupção utilizadas para indicar o início e fim das transmissões `interrupt_tx_begin` e `interrupt_tx_end`, as quais são geradas no

módulo da Figura 47. As 2 saídas `interrupt_sync_loss` e `interrupt_sync_achieved` são utilizadas para interromper a aplicação em ARM, indicando que o sistema perdeu ou recuperou sincronia com o contador de referência.

O bloco conta com a saída `ncr_out_reg` a qual é utilizada para monitorar a diferença entre o contador local e o contador remoto e com a saída `ncr_out_remote`, a qual retorna o valor do contador remoto.

O bloco *SINCRONIZE* da figura 49 possui os seguintes componentes sendo desenvolvido tendo como base o diagrama da figura 42.

4.3.1 ADPLL

O bloco recebe 2 contadores de entrada e realiza a sincronização dos mesmos. A sincronia é feita a partir da alteração da saída de offset de fase, a qual modifica a frequência do sinal de clock. A figura 51 apresenta as entradas e saídas desta entidade.

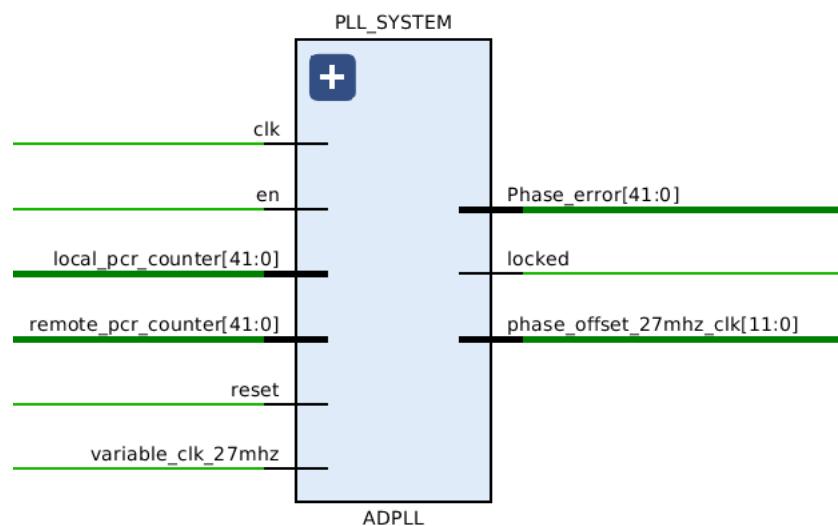


Figura 51 – Interface RTL do bloco de sincronia

4.3.2 Contador Local

O contador local possui entradas para carregar um novo valor de contagem, zerar a contagem e setar o valor máximo que o contador deve atingir antes de iniciar a contagem novamente

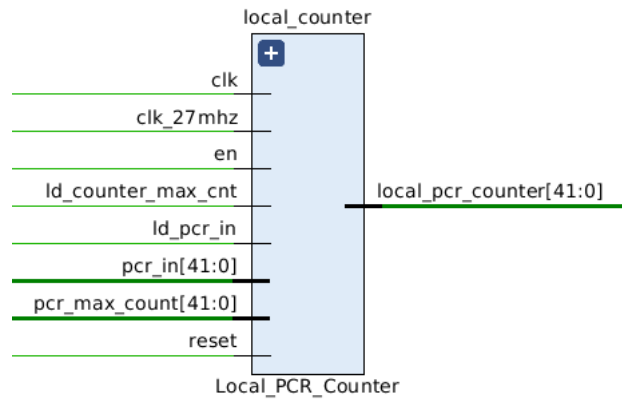


Figura 52 – Interface RTL do contador local

4.3.3 Detector de fase

Obtém a diferença entre o contador local e o contador remoto. A sua saída é conectada ao filtro IRR, para eliminar *jitter*.

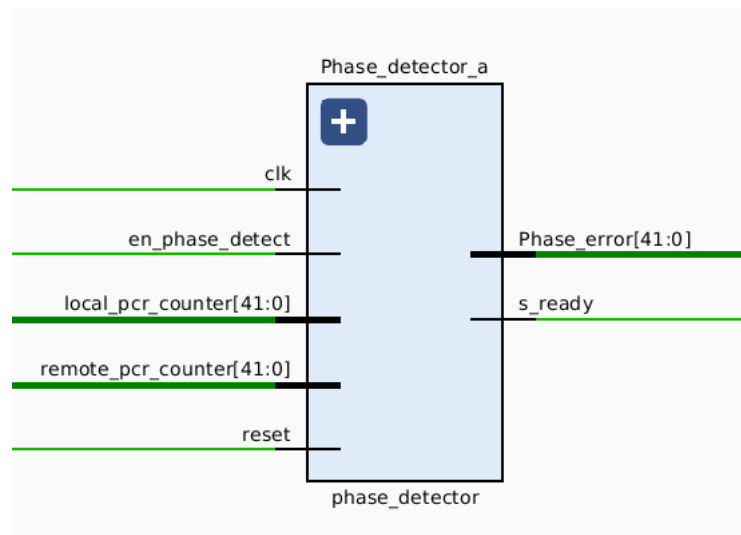


Figura 53 – Interface RTL do detector de fase

4.3.4 Filtro IRR

Este módulo recebe a saída do módulo da figura 53 e realiza a sua filtragem. O sinal filtrado é utilizado pelo módulo da figura 56 para ajustar a frequência do contador

de 27MHz.

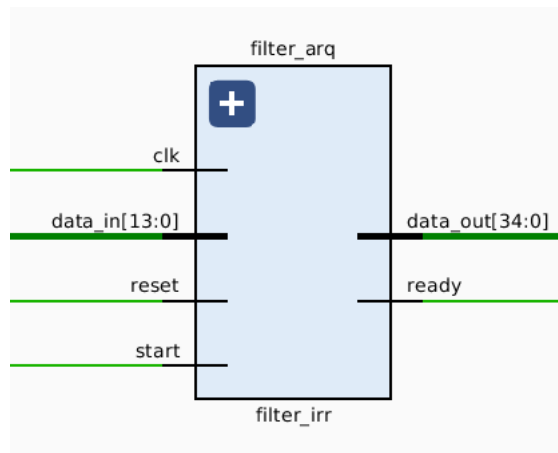


Figura 54 – Interface RTL do filtro IRR

4.3.5 Palavra de Frequência

Este bloco recebe o sinal de erro filtrado e o converte para um valor de ajuste de frequência. Este ajuste é aplicado ao módulo DDS, o qual é responsável por controlar a frequência de 27Mhz.

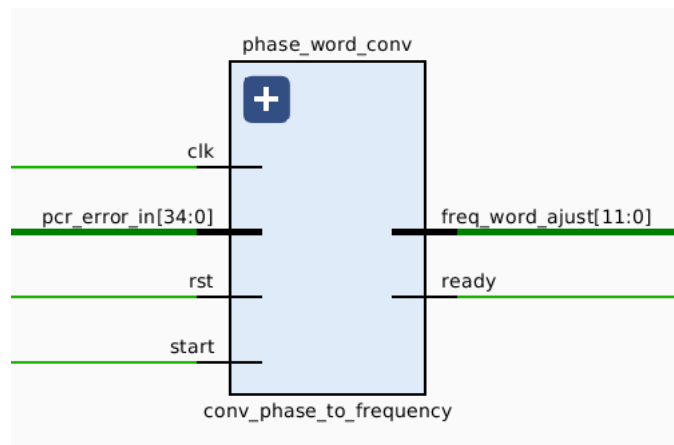


Figura 55 – Interface RTL do contador local

4.3.6 DDS

O módulo DDS gera um clock em torno de 27Mhz. Este valor é alterado para permitir que o sistema PLL da figura 51 funcione adequadamente, pois permite que o clock de 27Mhz possa ser alterado em pequenos passos.

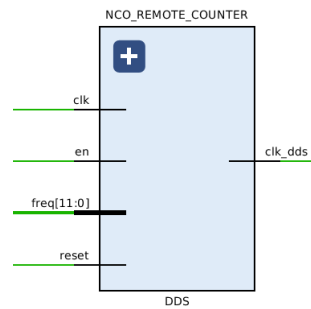


Figura 56 – Interface RTL do contador local

4.3.7 Arquitetura de teste

A validação do bloco de sincronia foi feita a partir da capacidade do sistema em realizar a sincronia com um contador extra implementado no sistema. Este contador possui um clock fixo de 27 MHz sendo referido como de contador remoto. A figura 57 apresenta a arquitetura proposta para a validação do sincronizador.

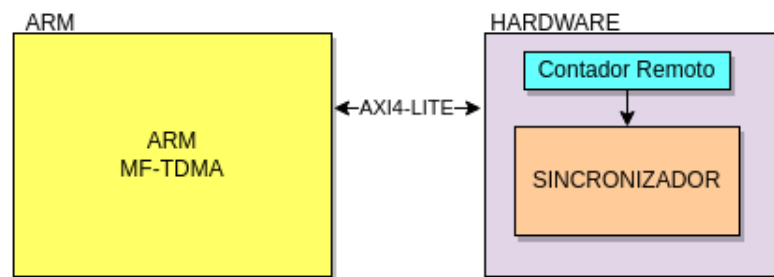


Figura 57 – Arquitetura de teste do sincronizador

A validação do sistema em ARM foi feita utilizando o barramento AXI-Lite para controlar e modificar o contador remoto. Após o início do sistema, o contador remoto foi desabilitado utilizando o barramento AXI4-Lite. Poderá ser observado na porta serial uma mensagem indicando que a sincronia foi perdida. Após 5 segundos do início do teste, foi solicitado ao sistema que realize a leitura dos registradores que armazenam o erro entre o contador local e o contador remoto. O valor lido deve diferir de zero, indicando que os 2 contadores estão fora de sincronia.

Após ser observado que os contadores estão fora de sincronia, foi indicado ao sistema para realizar a sincronia, e deve ser observado uma mensagem na porta serial indicando que a sincronia foi recuperada. Visando confirmar que os 2 contadores estão em sincronia, foi escrito na porta serial o valor obtido entre a diferença entre os 2 contadores, a qual deve ser menor ou igual a 0.

5 Resultados

5.1 simulação dos moduladores

5.1.1 Sinais em fase e quadratura filtrados

Para a implementação dos moduladores em ARM, um modelo em Octave foi implementado para servir de referência. Esses resultados foram numericamente comparados e serão apresentados a seguir para cada uma das modulações presentes no protocolo DVB-RCS2.

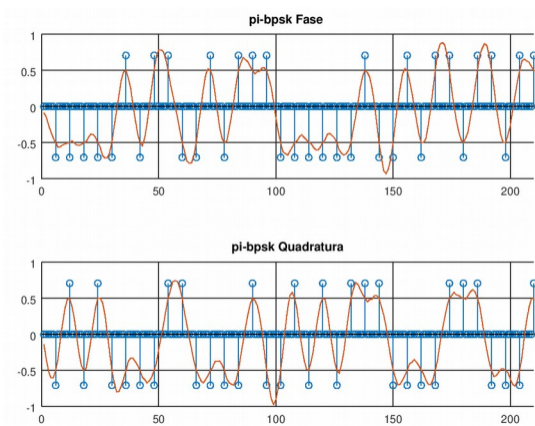


Figura 58 – octave: $\pi/2$ bpsk

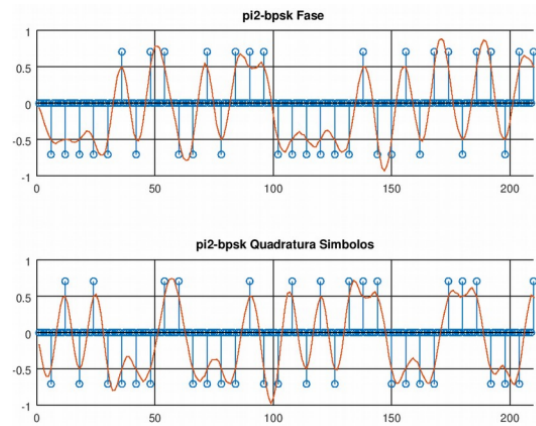


Figura 59 – ARM: $\pi/2$ BPSK

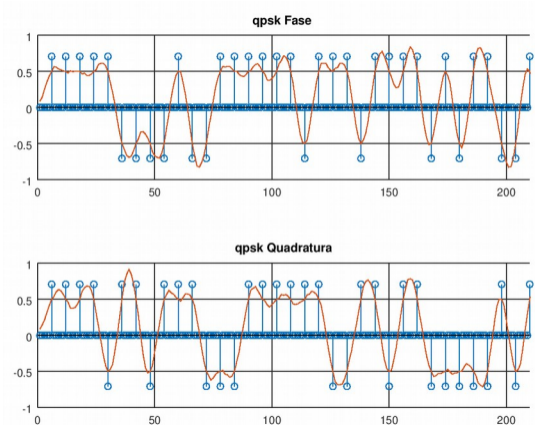


Figura 60 – octave: QPSK

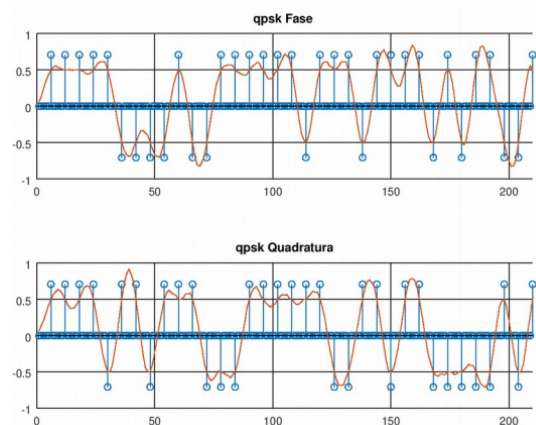


Figura 61 – ARM: QPSK

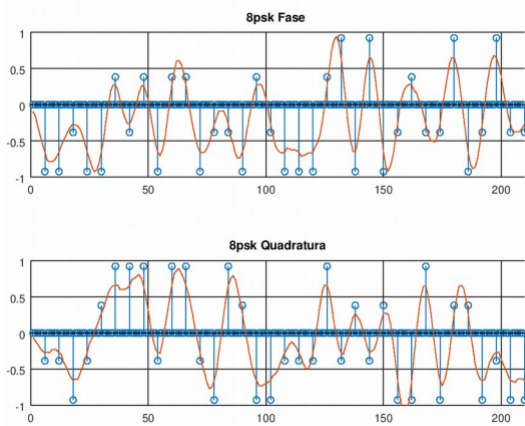


Figura 62 – octave: 8PSK

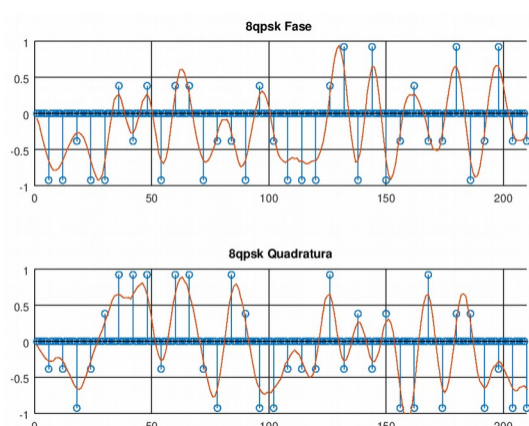


Figura 63 – ARM: 16QAM

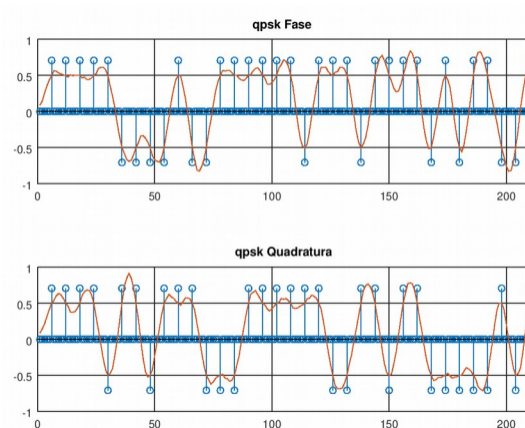


Figura 64 – octave: 16QAM.

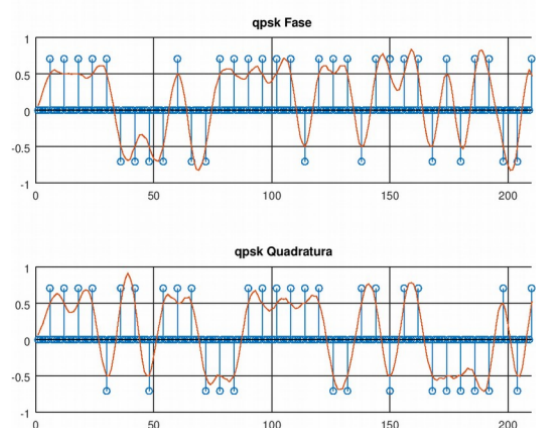


Figura 65 – ARM: 16QAM

As figuras 58 a 65 apresentam o resultado dos blocos de modulação após o processo de filtragem pelo filtro *square-root-raised-cosine filter* (SRRC). O sistema foi testado considerando uma entrada 220 amostras. Os gráficos estão em pares, de forma que um apresenta o modelo em octave, enquanto o outro apresenta o resultado obtido pela implementação em ARM do módulo.

A análise destes gráficos permite inferir que o ganho do filtro em ambas as implementações é de 1, e que o formato gerado é o mesmo para ambos os modelos. Desta forma, é possível validar a implementação em ARM com a implementação em Octave dos moduladores.

5.1.2 Diagrama de olho

A seguir será apresentado o diagrama de olho das simulações em Octave dos moduladores implementados. Os valores foram obtidos pelo modelo, e em seguida plotado utilizando a ferramenta do próprio sistema.

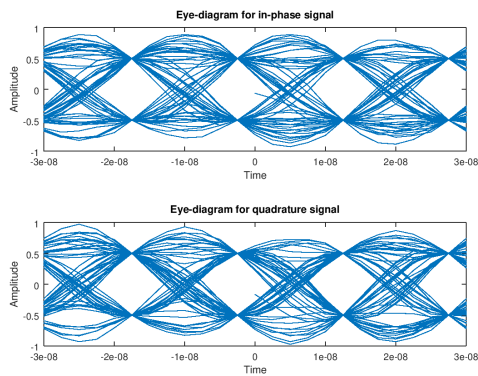


Figura 66 – Diagrama de olho: $\pi/2$ BPSK

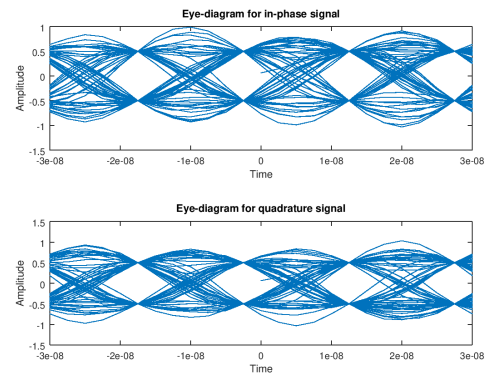


Figura 67 – Diagrama de olho: QPSK

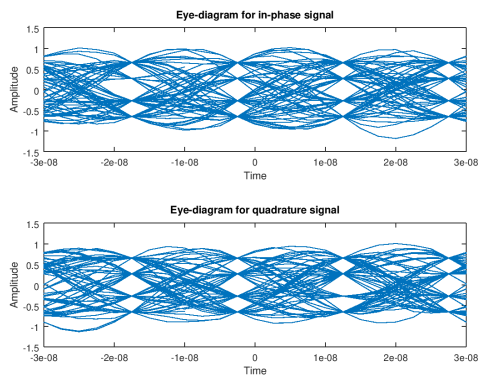


Figura 68 – Diagrama de olho: 8PSK

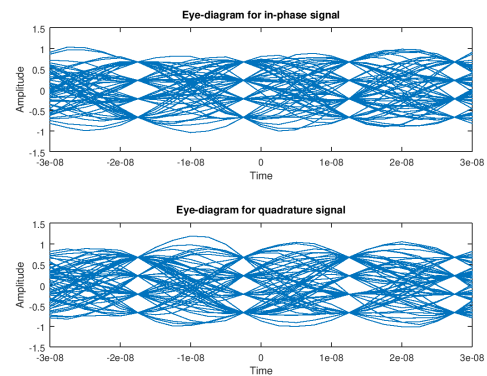


Figura 69 – Diagrama de olho: 16QAM

As figuras 66 a 69 apresentam o diagrama de olho para as modulações $\pi/2$ bpsk, QPSK, 8PSK e 16QAM. Analisando as figuras, é possível verificar que a Interferência entre Símbolos (ISI), é praticamente nula, pois no ponto de amostragem a distorção de amplitude presentes nas figuras são praticamente nulas.

Ademais, o melhor momento para realizar a amostragem do sinal, consiste no ponto em que essas sobreposições possuem para cada amostra, o mesmo valor de amplitude, e este ponto pode ser visualizado nas figuras.

5.1.3 Constelação

A seguir serão apresentadas as constelações obtidas das simulações em Octave do modelo dos moduladores. A obtenção das constelações consistiu em realizar um down sampling de fator L no sinal filtrado, desta forma, são obtidas apenas as amostras desejadas do sistema.

As figuras 70 a 73 apresentam corretamente as constelações das modulações utilizadas. A figura 70 apresentou 4 pontos na constelação, o que difere da constelação de um

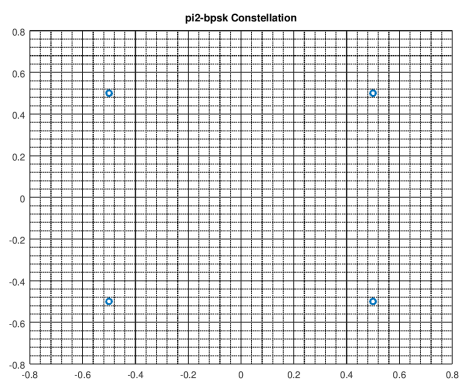


Figura 70 – Constelação: $\pi/2$ BPSK

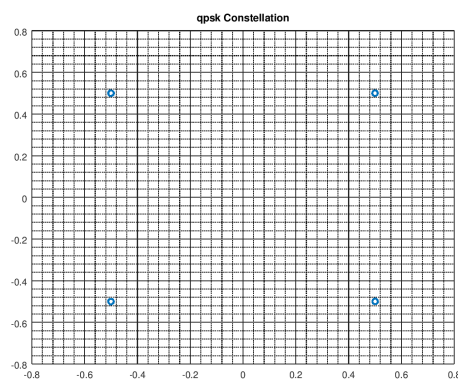


Figura 71 – Constelação: QPSK

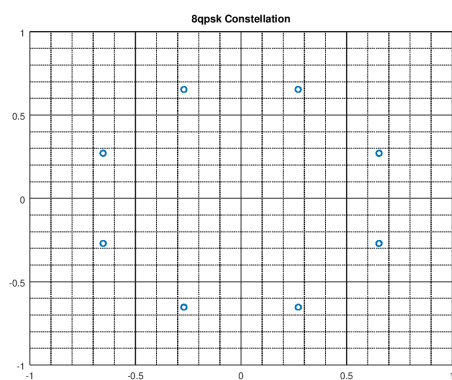


Figura 72 – Constelação: 8PSK

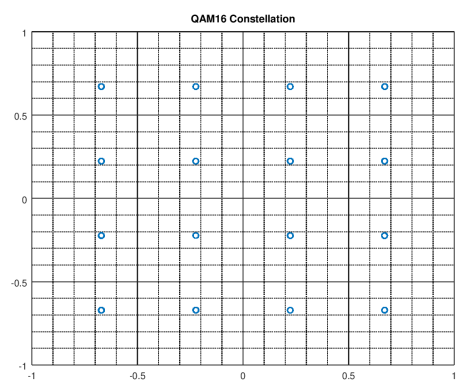


Figura 73 – Constelação: 16QAM

BPSK padrão, que possui apenas 2 pontos. Entretanto, essa diferença se deve ao tipo de BPSK utilizado no sistema, o qual consiste no no BPSK padrão rotacionado em 45 graus.

5.2 Simulação comportamental do bloco de transmissão

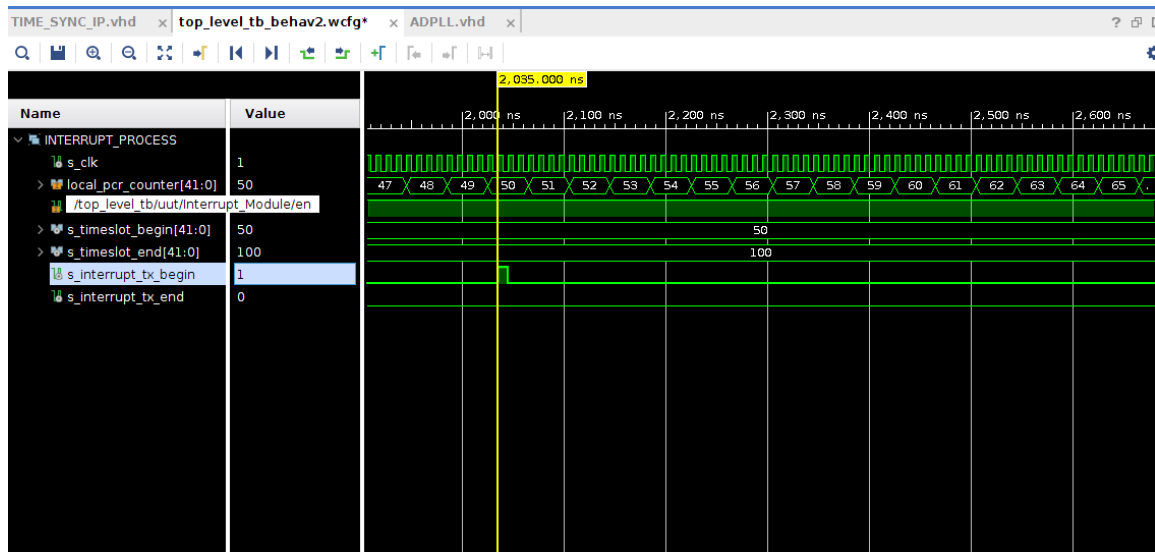


Figura 74 – Transmissão: Tx Begin

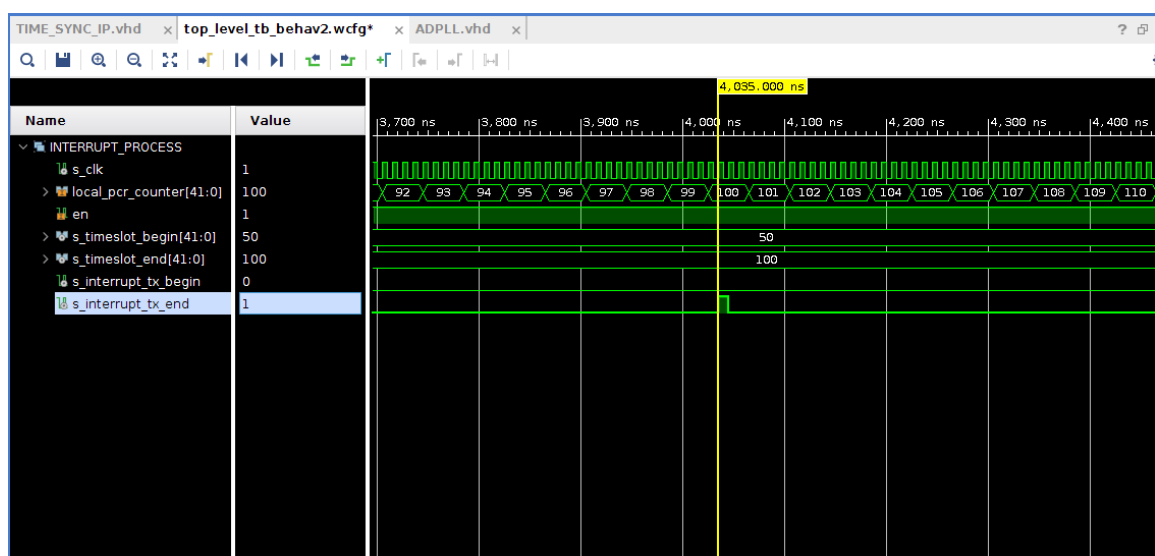


Figura 75 – Transmissão: Tx End

As figuras 74 e 75 apresentam a simulação comportamental no Vivado do estado TRANSMIT da figura 49. Na simulação é possível verificar que o sinal `s_timeslot_begin` esta configurado em 50. No momento em que o sinal `local_pcr_counter` atinge este valor, o sinal de saída `s_interrupt_tx_begin` altera de 0 para 1, durante um ciclo de relógio. De forma semelhante, na figura 75 é possível verificar que o mesmo ocorre para o sinal `s_timeslot_end`, gerando um pulso na saída `s_interrupt_tx_end`. Este bloco é responsável por indicar o momento das transmissões do MF-TDMA e apenas esta habilitado no estado de TRANSMIT.

5.3 Simulação do bloco de sincronização

A simulação do bloco de sincronia foi feito a partir do Vivado conforme se mostra a seguir.



Figura 76 – Sincronizador: Sem sincronia A

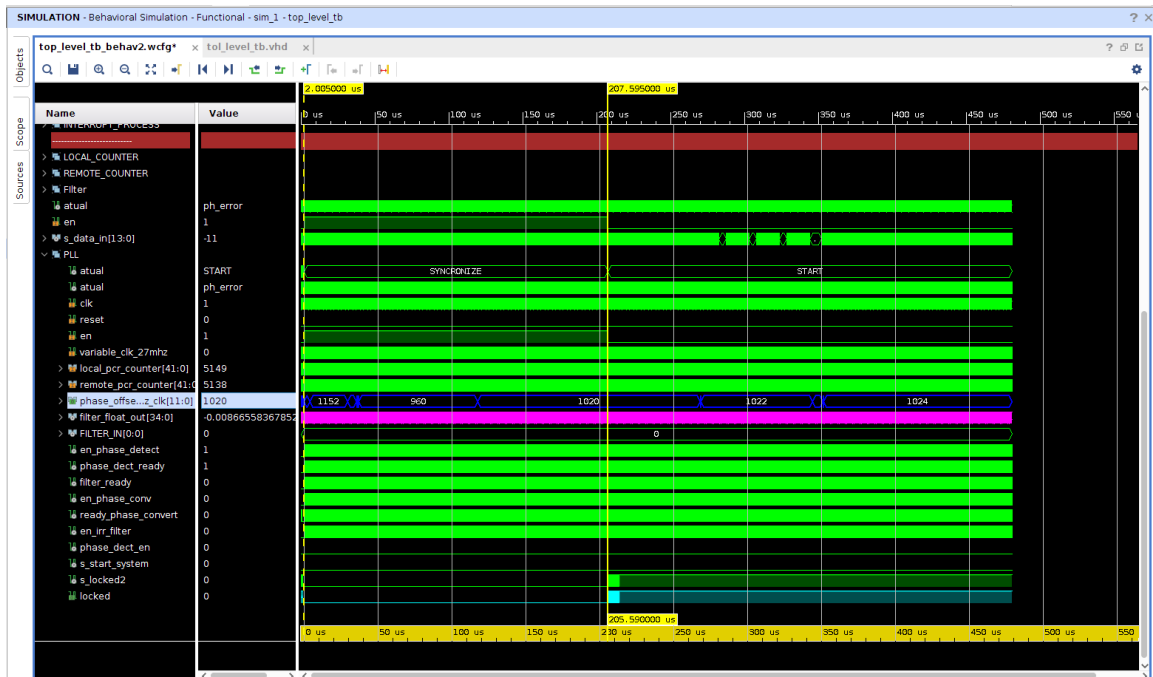


Figura 77 – Sincronizador: Sincronia recuperada A

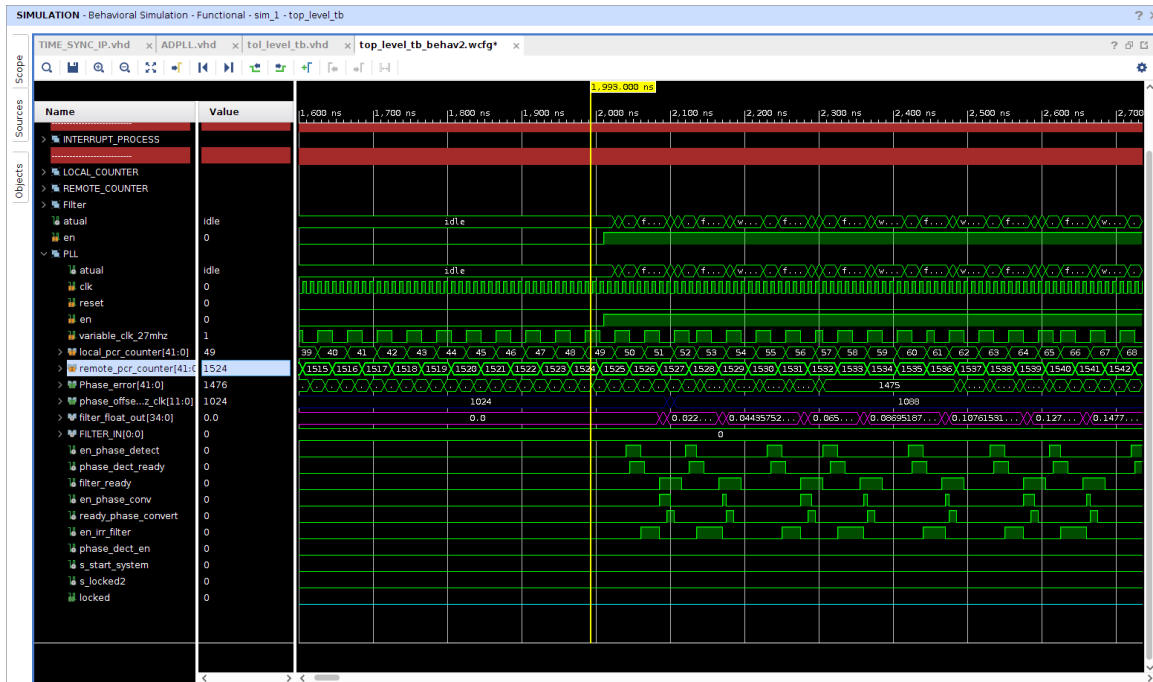


Figura 78 – Sincronizador: Sem sincronia B



Figura 79 – Sincronizador: Sincronia recuperada B

As simulações presentes nas figuras 76 e 78 foram feitas alterando o valor de contagem do contador remoto, de forma a forçar a dessincronia do sistema. Isto pode ser observado nas imagens, sobre o cursor, no tempo 1,955 ns e 1,933 ns, respectivamente. O sinal locked esta igual a zero, indicando que o sistema esta sem sincronia. O erro entre o contador local e o contador remoto pode ser observado na figura como o nome Phase_Error com os valores 675 e 1517 nas figuras 76 e 78. O sinal en será habilitado

para que o sistema possa realizar a sincronia dos contadores fazendo com que o sistema saia do estado de START para o estado de SYNCRONIZE.

As figuras 77 e 79 apresentam o momento em que a sincronia dos 2 contadores foi recuperada. É possível verificar na imagem que em 207.59 us, o primeiro pulso do sinal locked aparece, indicando que o sistema está sincronizando. Passado alguns pulsos, o sinal locked permanece sempre em 1, indicando que o sistema está com os 2 contadores sincronizados. É possível observar que o sinal `Phase_offset` apresentado em azul nas simulações, sofre alterações conforme o decorrer da simulação. Este valor é utilizado para modificar a frequência do contador local, o qual é apresentado na imagem como `local_pcr_counter`.

5.4 Verificação da integração RCS2: *Modulador, Spreading, filtro*

A implementação integração dos módulos spreading, filtros e moduladores foram elaboradas em linguagem C, utilizando a ferramenta SDK 2018. Os resultados foram obtidos considerando a cadeia completa do codificador DVB-RCS2.

A seguir será apresentada de maneira gráfica a sobreposição entre o resultado obtido na implementação em ARM, com o modelo em software em python executado em um PC de escritório.

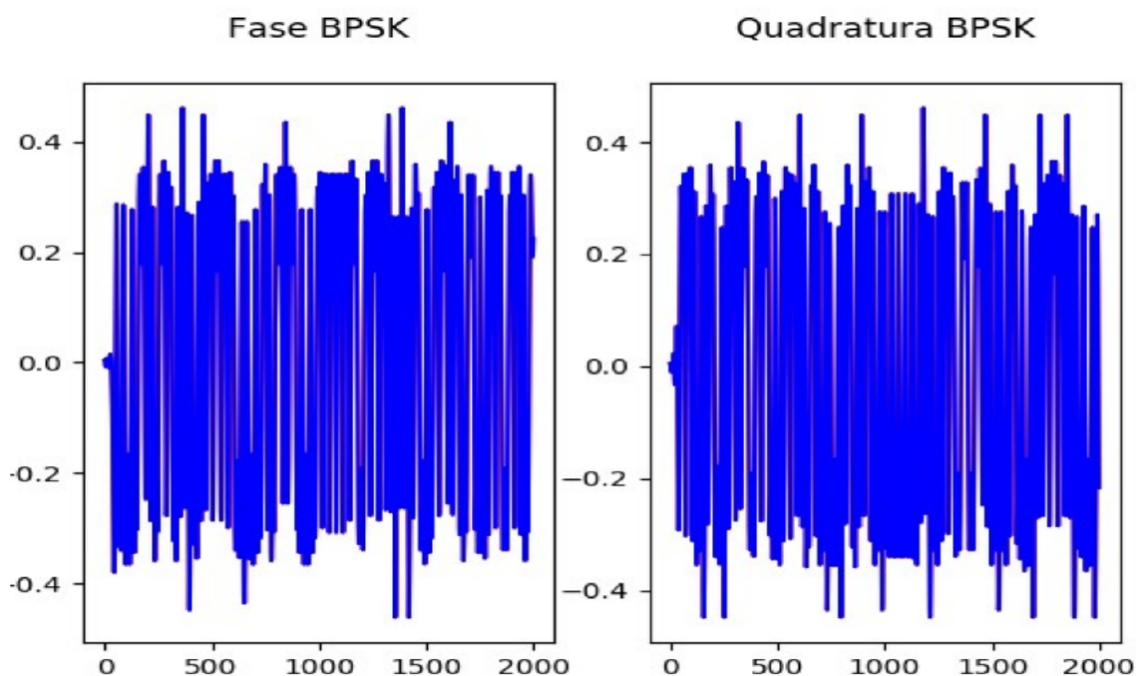


Figura 80 – Formas de ondas sobrepostas da implementação em ARM com modelo em python - BPSK

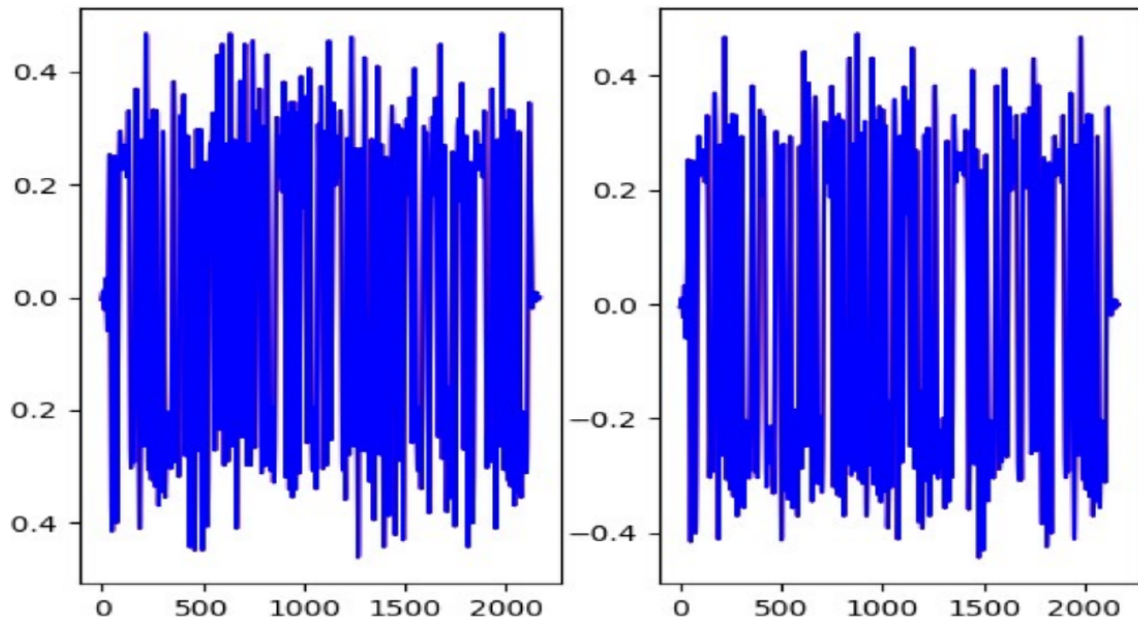


Figura 81 – Formas de ondas sobrepostas da implementação em ARM com modelo em python - QPSK

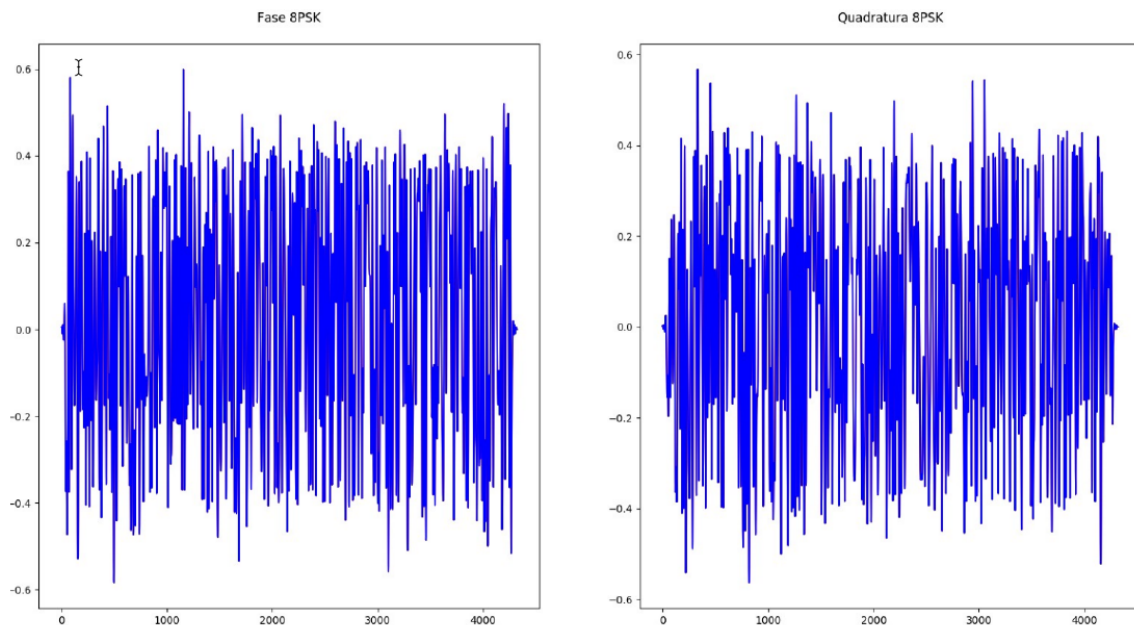


Figura 82 – Formas de ondas sobrepostas da implementação em ARM com modelo em python - 8PSK

As figuras 80, 81 e 82 expõem a sobreposição do modelo em python com a implementação em ARM. Como pode ser visto nas figuras, as formas de onda estão completamente sobrepostas, confirmando que a implementação em ARM estão corretamente codificada.

Os resultados obtidos quanto ao tempo de execução do codificador RCS2 para cada uma das formas de ondas presente no protocolo DVB-RCS2 será apresentada a seguir,

de forma que os resultados estão divididos 2 partes. *Spread-spectrum Linear* (SSLM), e *Turbo Code Linear Modulation* (LM). Para cada das *Wave ID* fora simulado e obtido o tempo execução em software do sistema.

| LM | | SSLM | |
|---------|-----------|---------|------------|
| Wave ID | Time(ms) | Wave ID | Time(ms) |
| 1 | 5,511825 | 1 | 29,272031 |
| 2 | 2,185704 | 2 | 21,94835 |
| 3 | 4,444128 | 3 | 49,903433 |
| 4 | 4,470747 | 4 | 36,165422 |
| 5 | 4,484451 | 5 | 125,006246 |
| 6 | 4,5231 | 6 | 93,761493 |
| 7 | 4,531323 | 7 | 58,537657 |
| 8 | 4,506945 | 8 | 43,79492 |
| 9 | 4,558251 | 9 | 99,842565 |
| 10 | 4,570215 | 10 | 72,313306 |
| 11 | 4,647222 | 11 | 249,285656 |
| 12 | 4,664703 | 12 | 187,158298 |
| 13 | 13,350192 | 13 | 117,351363 |
| 14 | 13,440273 | 14 | 87,842412 |
| 15 | 13,451685 | 15 | 199,60095 |
| 16 | 13,592814 | 16 | 144,784589 |
| 17 | 13,620075 | 17 | 497,883541 |
| 18 | 13,508964 | 18 | 373,527841 |
| 19 | 13,718802 | 19 | 126,347159 |
| 20 | 13,760379 | | |
| 21 | 13,99671 | | |
| 22 | 14,050227 | | |
| 32 | 6,899778 | | |
| 33 | 4,775007 | | |
| 34 | 11,514495 | | |
| 35 | 7,907013 | | |
| 36 | 6,847299 | | |
| 37 | 22,144085 | | |
| 38 | 18,582221 | | |
| 39 | 17,536364 | | |
| 40 | 15,324954 | | |
| 41 | 13,235733 | | |
| 42 | 26,622215 | | |
| 43 | 26,683025 | | |
| 44 | 2,269845 | | |
| 45 | 2,251212 | | |
| 46 | 2,273913 | | |
| 47 | 2,27679 | | |
| 48 | 2,318694 | | |
| 49 | 2,323251 | | |

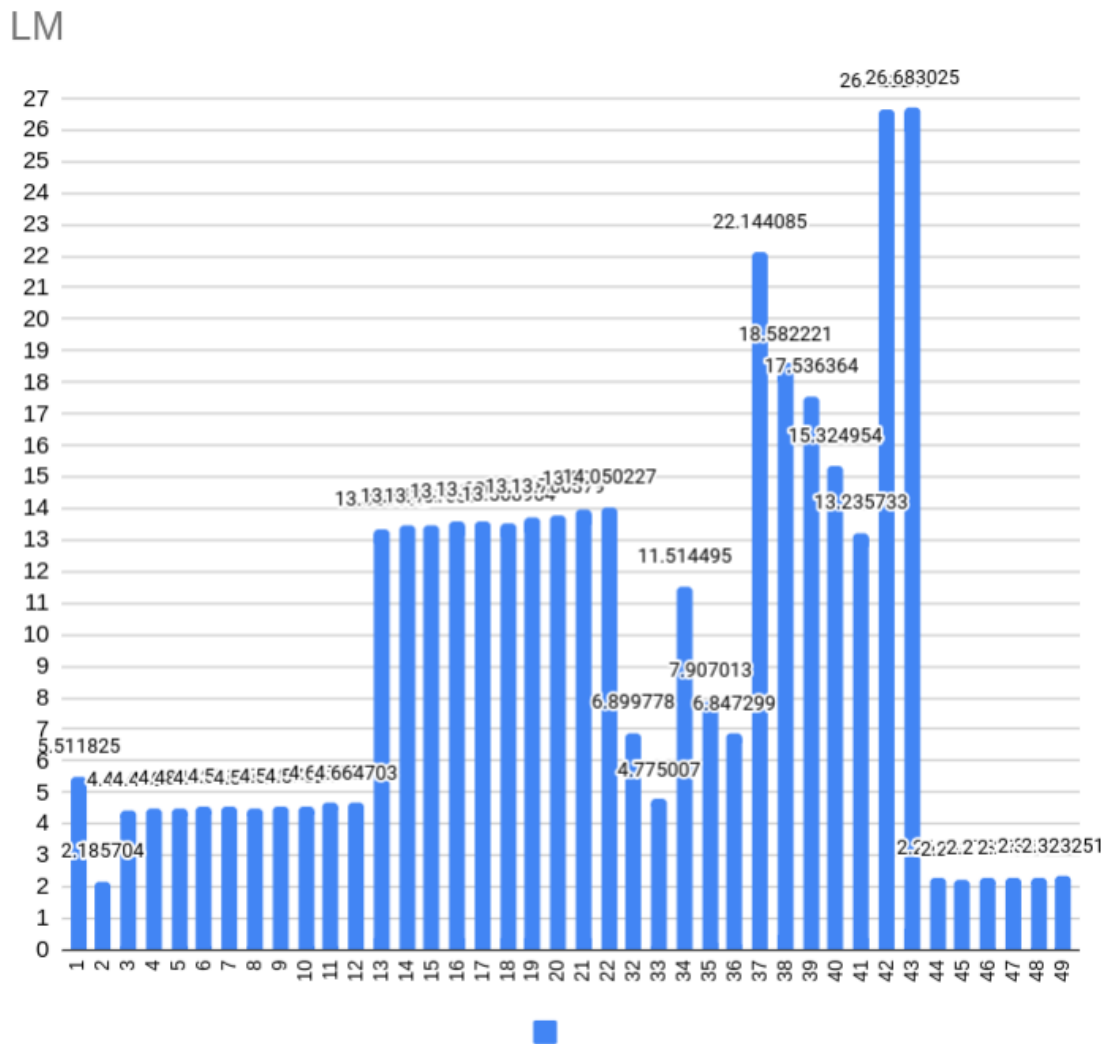


Figura 83 – Tempo de execução: *Linear Modulation*

A figura 83 apresenta de forma gráfica os resultados de tempo de execução considerando o TC-LM. Os resultados apresentados permitem inferir que as Wave ID com maior tempo de execução são as de número 42 e 43, com valores em torno de 26,7ms. O aumento do tempo de execução esta ligado ao tipo de modulação utilizada, fator de sobre amostragem L. Desta forma, um maior fator de sobreamostragem grande gera um *Burst* maior, o que aumenta o tempo de execução para a Wave ID.

O tempo de execução obtido para cada Wave ID está dentro do estabelecido pela norma DVB-RCS2, a qual prevê que o valor máximo para o tempo de execução deve ser menor que 40ms.

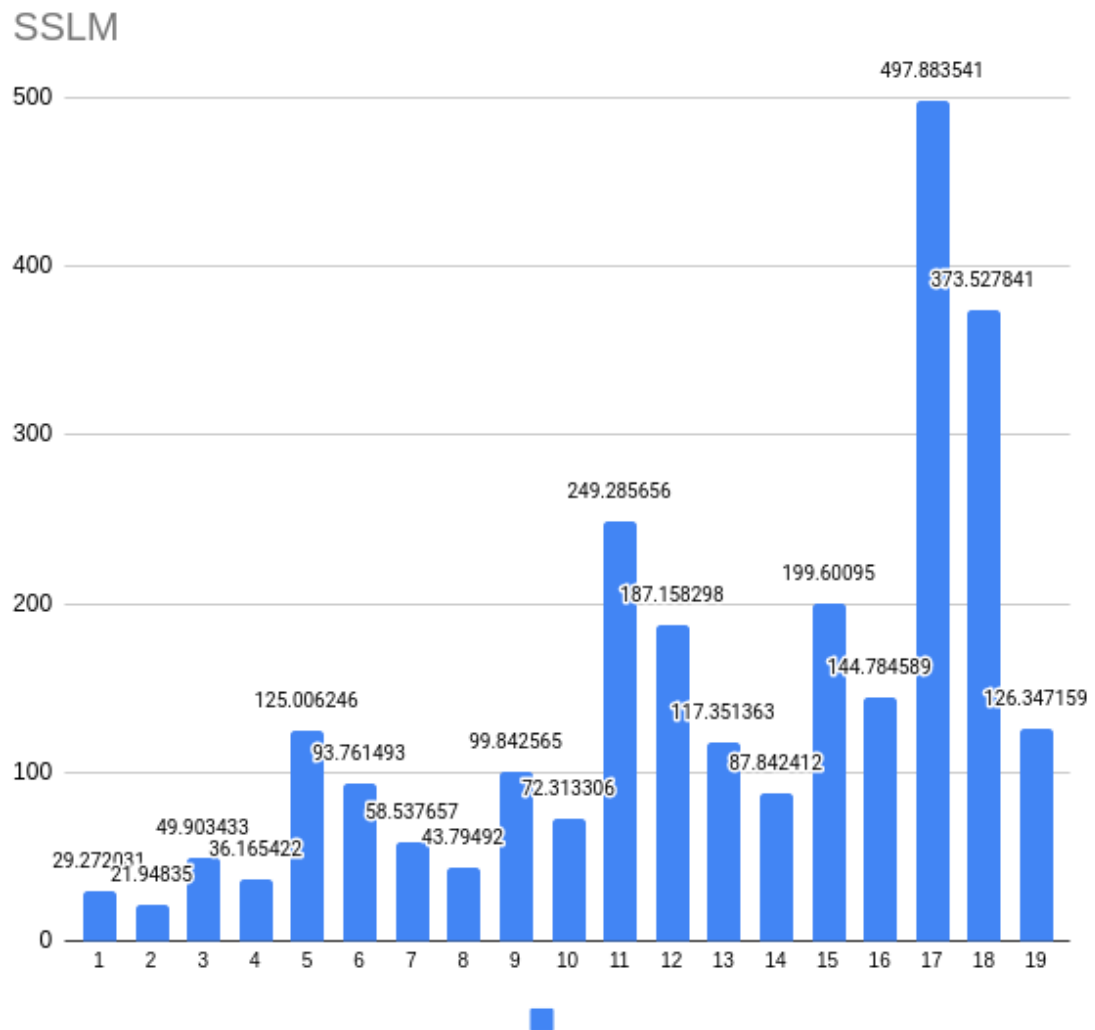


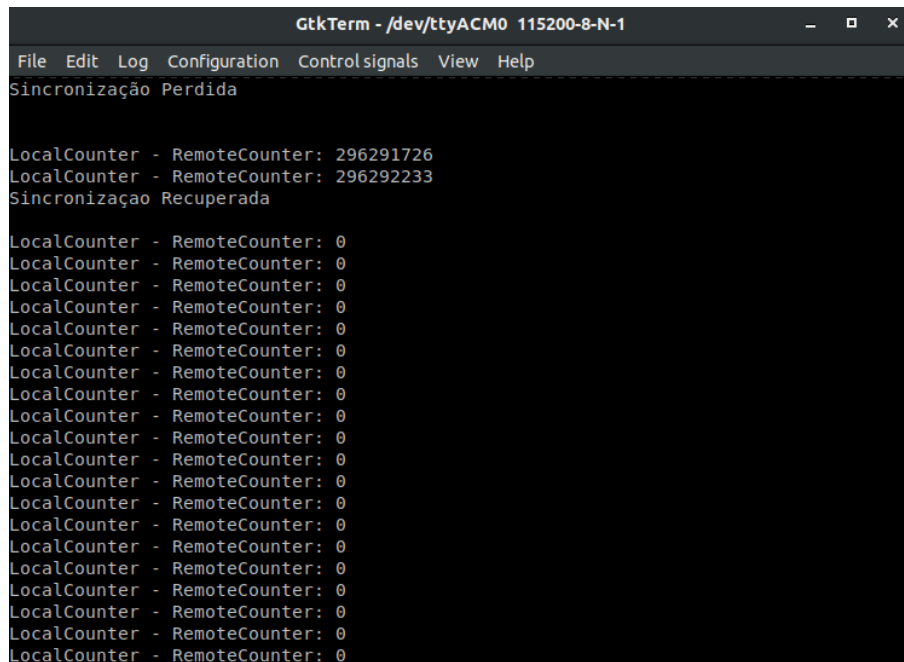
Figura 84 – Tempo de execução: *Spread Spectrum Linear Modulation*

A figura 84 apresenta de forma gráfica os resultados de tempo de execução considerando o SS-LM. Os resultados obtidos apresentaram um tempo de execução maior que o do sistema TC-LM, de forma que o maior tempo de execução corresponde a *wave ID* 17, a qual apresentou um tempo de execução de 497 ms. O sistema SS-LM possui, em geral um maior tempo de execução, o qual é explicado pelo tamanho do *Burst* gerado ao final do processo de *Spread*.

Os sistemas com espalhamento de espectro (SS-LM) são mais robustos, entretanto são mais lentos. Isto se deve ao tamanho dos *bursts* gerados devido ao tipo de codificação utilizado. A *wave ID* 17 apresenta tamanho de 7548 símbolos, sendo a maior no sistema.

5.5 Sincronizador

A figura 85 apresenta o caso, em que em algum estado, a HUB indicou ao sistema que ocorreu perda de sincronismo. Neste estado foi feita a recuperação do sincronismo



```
GtkTerm - /dev/ttyACM0 115200-8-N-1
File Edit Log Configuration Controlsignals View Help
Sincronização Perdida
LocalCounter - RemoteCounter: 296291726
LocalCounter - RemoteCounter: 296292233
Sincronização Recuperada
LocalCounter - RemoteCounter: 0
LocalCounter - RemoteCounter: 0
LocalCounter - RemoteCounter: 0
LocalCounter - RemoteCounter: 0
LocalCounter - RemoteCounter: 0
LocalCounter - RemoteCounter: 0
LocalCounter - RemoteCounter: 0
LocalCounter - RemoteCounter: 0
LocalCounter - RemoteCounter: 0
LocalCounter - RemoteCounter: 0
LocalCounter - RemoteCounter: 0
LocalCounter - RemoteCounter: 0
LocalCounter - RemoteCounter: 0
LocalCounter - RemoteCounter: 0
LocalCounter - RemoteCounter: 0
LocalCounter - RemoteCounter: 0
LocalCounter - RemoteCounter: 0
LocalCounter - RemoteCounter: 0
```

Figura 87 – Sincronizador: Perda de sincronia após 10 segundos

As figuras 86 e 87 apresentam o experimento para validação do bloco de sincronismo. No início do experimento foi desabilitado o contador remoto por um período de tempo, e isto levou a perda de sincronia entre o contador local e o contador remoto. O experimento da figura 86 gerou um erro entre os 2 contadores em 5 segundos, enquanto o experimento da figura 87 gerou um erro de 10 segundos entre os contadores.

É possível verificar que é apresentado na imagem a informação "Sincronização Perdida". Nas imagens é possível observar que a diferença entre o contador local e o contador remoto antes de recuperar a sincronia estão com um valor diferente de zero, o que demonstra que os contadores estão fora de sincronia. Após isto, foi enviada a informação para realizar a sincronia do sistema. A mensagem Sincronia recuperada é apresentada logo em seguida, indicando que o sistema recuperou a sincronia entre os 2 contadores. Por fim, é apresentada algumas vezes o valor da diferença entre os contadores, para apresentar que a sincronia foi realmente recuperada. Sendo esta diferença igual a zero.

5.5.1 Consumo de recursos de hardware do módulo sincronizador

O Bloco sincronizador foi implementado inteiramente em hardware, e embora trabalhe em associação com o processador ARM, necessária realizar uma análise do consumo de recursos de hardware. A figura 88 apresentam o consumo de recursos utilizados na implementação do módulo de sincronização. É possível observar na figura 88 que o bloco que mais utiliza recursos da FPGA é o módulo ADPLL, o qual possui a arquitetura apresentada em 51. A maioria dos recursos são destinados á implementação do filtro IRR deste bloco.

| Name | Slice LUTs (53200) | Slice Registers (106400) | Bonded IOB (200) | BUFGCTRL (32) |
|--|-----------------------|-----------------------------|---------------------|------------------|
| ∨ top_level | 2073 | 726 | 222 | 6 |
| Interrupt_Module (Timeslots_Interrupt_Signals) | 6 | 88 | 0 | 0 |
| local_counter (Local_PCR_Counter) | 378 | 168 | 0 | 0 |
| > NCO_LOCAL_COUNTER (DDS) | 15 | 25 | 0 | 0 |
| > NCO_REMOTE_COUNTER (DDS_0) | 2 | 7 | 0 | 0 |
| ∨ PLL_SYSTEM (ADPLL) | 1331 | 217 | 0 | 0 |
| filter_arq (filter_irr) | 1149 | 153 | 0 | 0 |
| Phase_detector_a (phase_detector) | 170 | 43 | 0 | 0 |
| phase_word_conv (conv_phase_to_frequency) | 6 | 11 | 0 | 0 |
| remote_counter (Local_PCR_Counter_1) | 336 | 168 | 0 | 0 |

Figura 88 – Sincronizador: Consumo de recursos

5.6 Verificação do *Multi-frequency time-division multiple access*

5.6.1 Resultados sem emulação de HUB

Os resultados de desempenho do MF-TDMA serão apresentados de duas formas. Inicialmente a lógica de transição de estados das FSMs do MF-TDMA será avaliada sem a presença de uma HUB. Em seguida, será emulada uma HUB usando o SDR Adalm Pluto e verificando a recuperação de sincronismo e a transmissão na frequência e tempo requeridos pela HUB.

Para fins de visualização, o sistema implementado em ARM, foi conectado ao software Labview e a partir de certas condições de teste foi possível verificar de forma gráfica a mudança estados, bem como visualizar dentro de um *frame*, o *timeslot* acessado. As figuras em verde ou amarelo representam uma atividade válida do sistema.

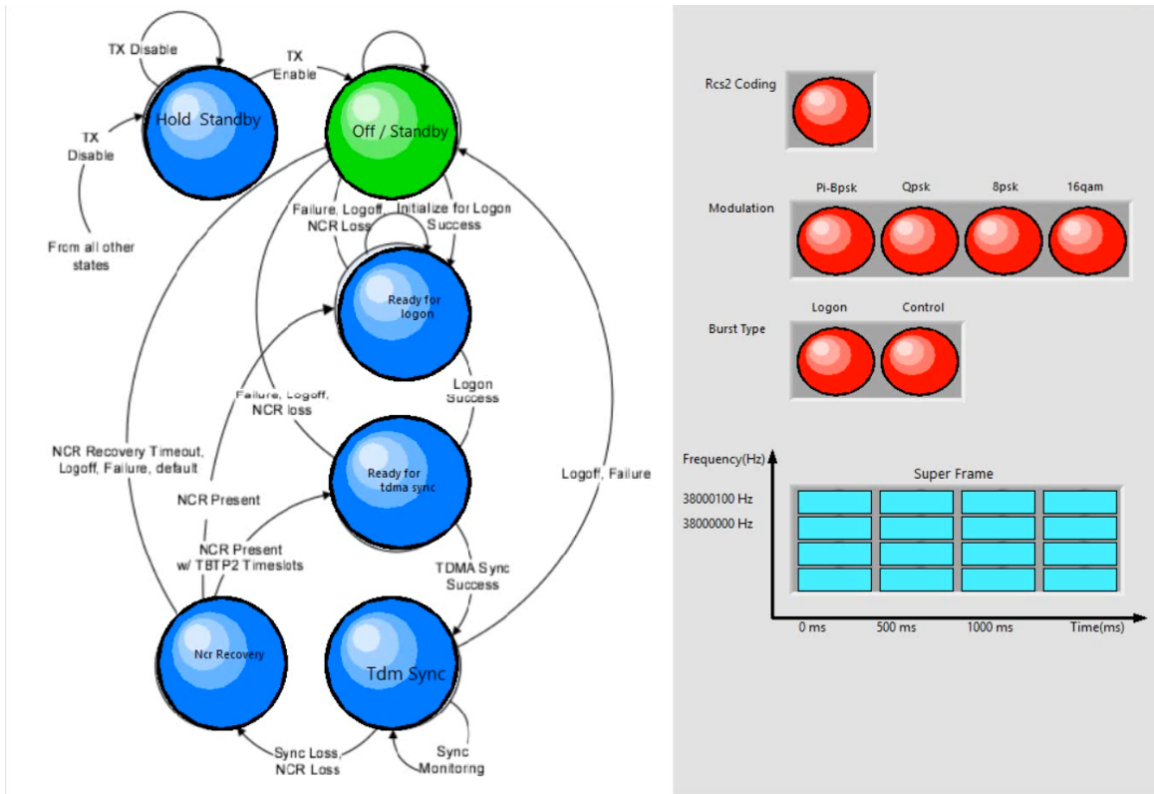


Figura 89 – MF-TDMA: Off/standby

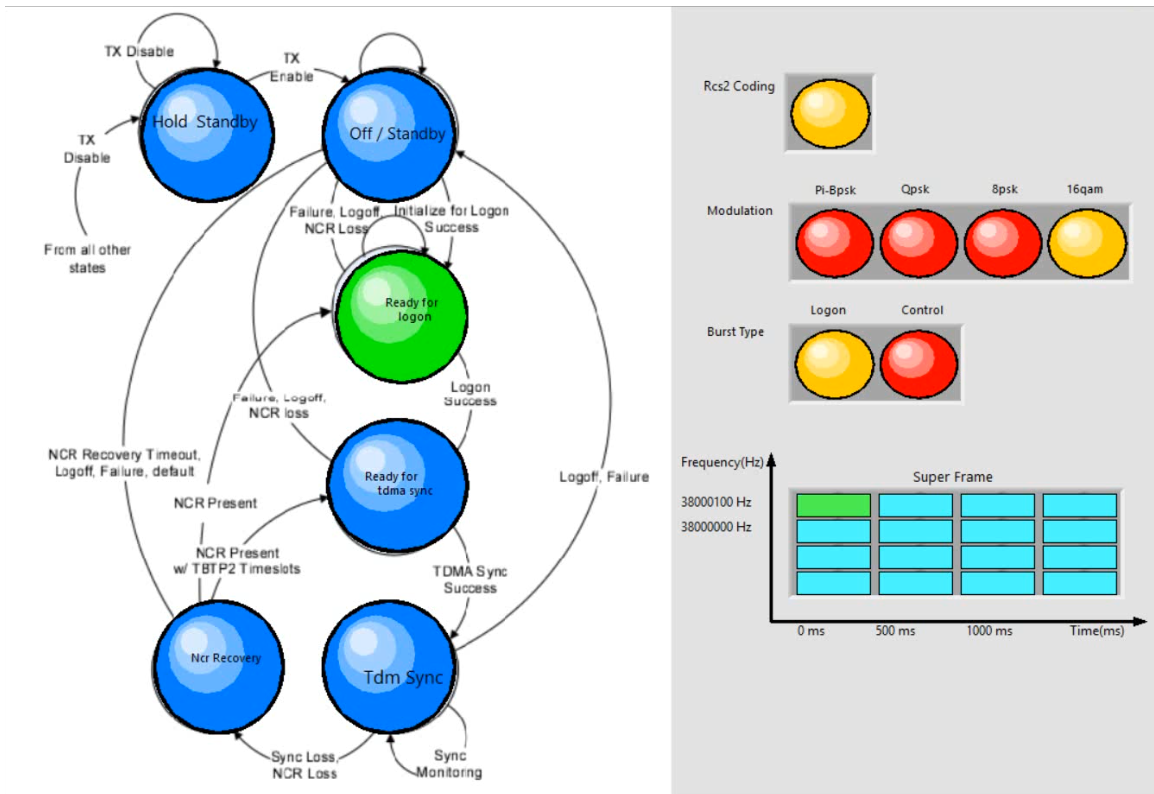


Figura 90 – MF-TDMA: Ready for logon

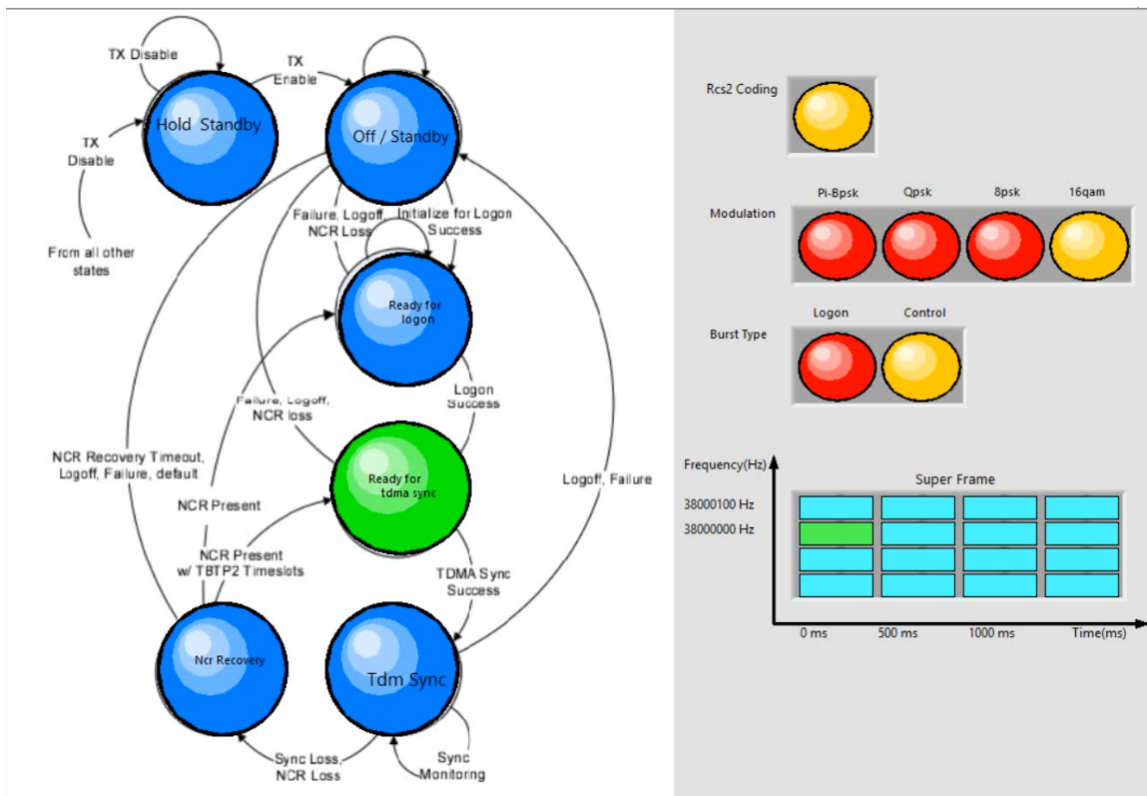


Figura 91 – MF-TDMA: Ready for tdma sync

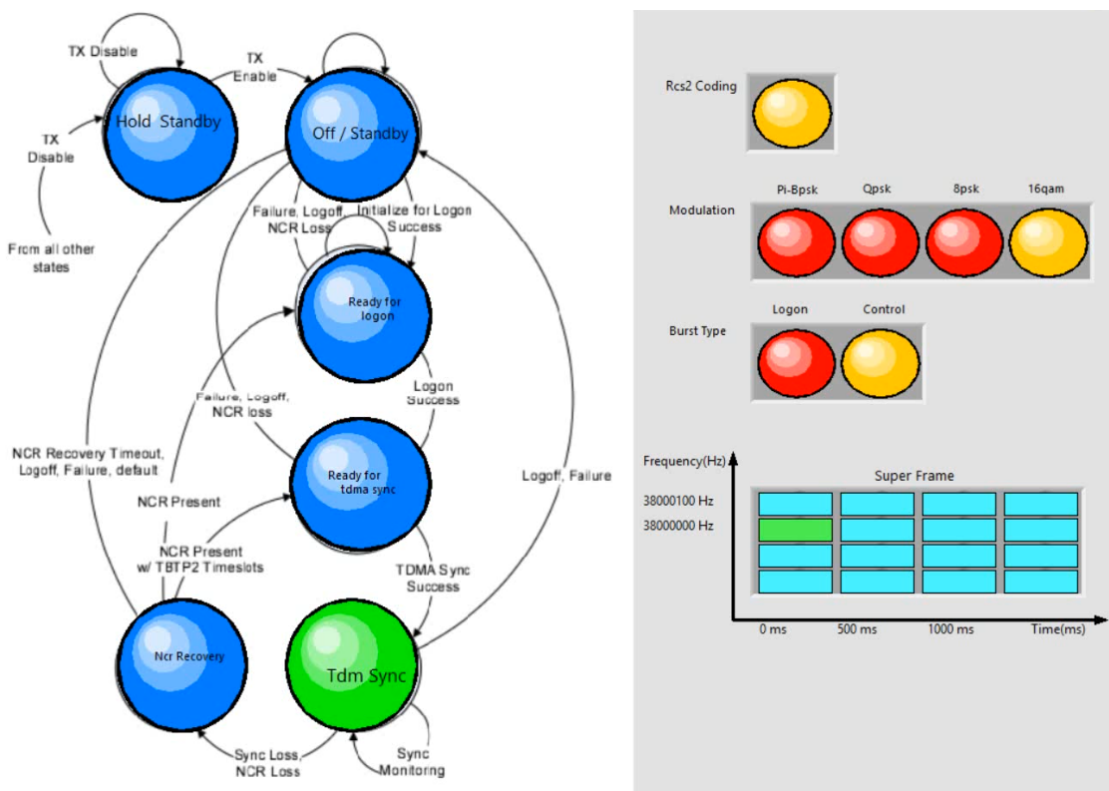


Figura 92 – MF-TDMA: tdma sync

As figuras 89 a 92 apresentam de forma gráfica o funcionamento do MF-TDMA. Cada gráfico apresenta um campo que indica que esta ocorrendo a codificação RCS2, a

modulação utilizada e o tipo de *burst* transmitido. Ademais, foi possível verificar o *timeslot* em que ocorre uma transmissão.

A figura 89 apresenta a situação em que o sistema estava em funcionamento. Neste estado são feitas as preparações básicas para uma tentativa de logon na rede. A implementação deste estado consistiu em apenas verificar as condições para validar uma iniciação válida de logon, de forma que as funções internas do estado não estão implementadas.

A figura 90 apresenta a situação em que o sistema conseguiu obter os recursos necessários para um logon. Neste estado, o sistema apresentou todo o algoritmo de logon que a norma DVB-RCS2 fornece. O estado apresenta as funções internas de codificação do sistema RCS2, obtenção de *timeslots* para acesso aleatório e verificação da resposta da HUB a partir das tabelas e descritores.

A figura 91 apresenta o cenário em que o sistema foi capaz de realizar um logon em acesso aleatório na rede. O sistema possui as funções de codificação do sistema RCS2, obtenção de *timeslots* de controle a partir das tabelas e descritores.

A figura 92 apresenta o último estado em que as funções internas foram executadas. As funções implementadas neste estado, são semelhantes as do estado da figura 91.

Para as figuras apresentadas o sistema realizou uma transmissão em um *timeslot* de logon com frequência de 38000100 Hz, e que inicia no tempo zero do *frame*. No estado de controle, o sistema realizou a transmissão em um *timeslot* para controle, o qual possui frequência de 38 MHz, que inicia no tempo zero do *frame*.

5.6.2 Resultados com emulação de HUB

Para fins de visualização, o sistema implementado em ARM, foi conectado a uma aplicação em python, de forma que dadas certas condições de teste, seja possível observar graficamente o estado atual em que o sistema se encontra, bem como as configurações atuais para transmissão dos *timeslots*. A máquina de estado de referência pode ser consultada na figura 37 e o vídeo do experimento pode ser acessado em (ALVES, a) e para demonstração das configurações dos *timeslots* em (ALVES, b).

A figura 93 apresenta o setup utilizado para a validação do teste do sistema. O Terminal consiste em SDR a partir de um kit Soc FPGA Zedboard e uma placa de interface FMCOMMS3 (baseado no transceiver AD93610) tendo o canal TX2 E RX2 habilitados para uso. A HUB foi emulada usando um PC de escritório, um SDR Adalm Pluto em conjunto com o software GNU Radio. A HUB realiza as alterações das tabelas e descritores do MF-TDMA, conforme apresentado nas tabelas 2 e 4.

A seguir serão apresentados os resultados obtidos para cada reconfiguração do MF-TDMA utilizando a HUB emulada. Serão observadas as mudanças de estado e a

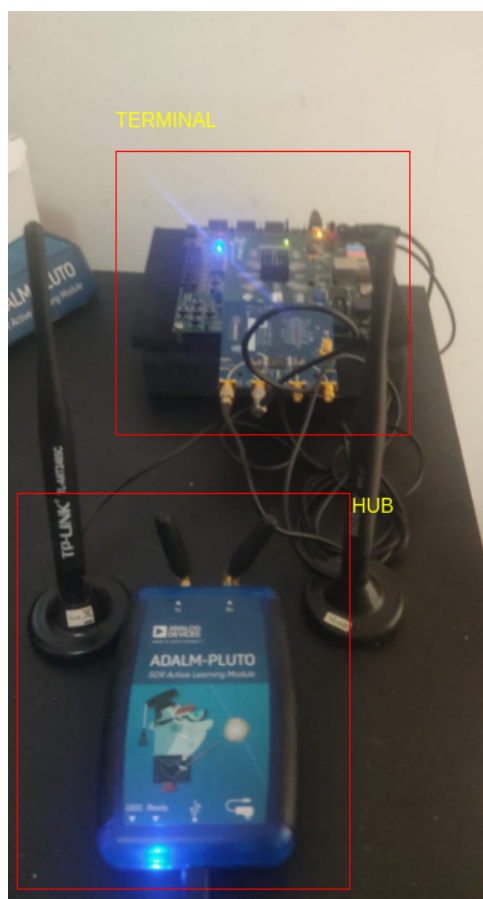


Figura 93 – Mf-TDMA: Setup experimental

configuração do timeslot utilizado para transmitir o sinal DVB-RCS2. Em todos os casos de teste, é utilizado um sinal DVB-RCS2 codificado em QPSK. Ademais, será apresentado a tela do GNU Radio no momento em que o mesmo identificou uma recepção na frequência requerida e no *timeslots* solicitado. As imagens do GNU Radio apresentam a recepção do sinal no domínio da frequência em banda base. O GNU Radio realiza a demodulação do sinal na banda passante para a banda base, desta forma todos os sinais estarão centrados na frequência zero. As configurações dos *timeslots* estão presentes na tabela 7.

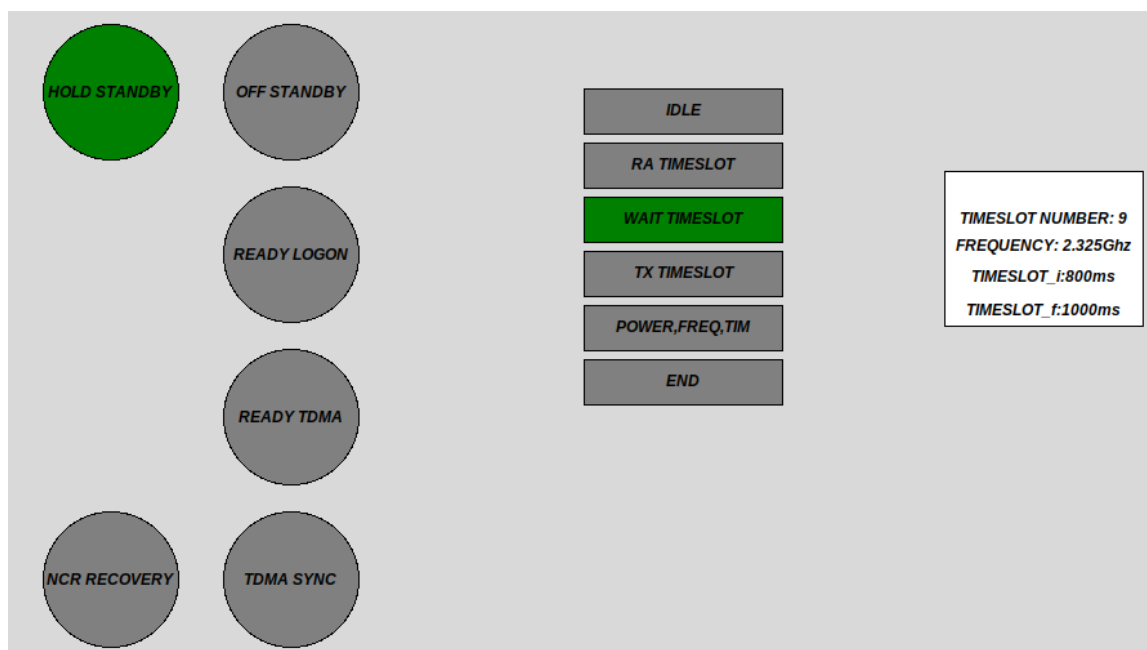


Figura 94 – MF-TDMA: Hold Standby

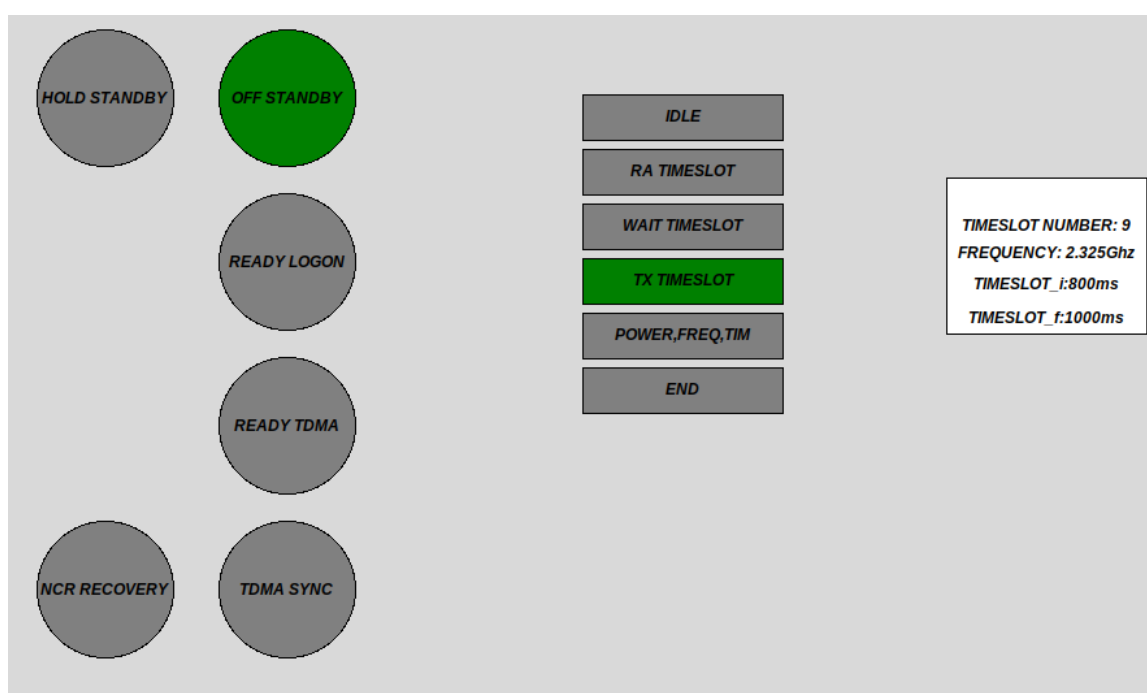


Figura 95 – MF-TDMA: Off Standby

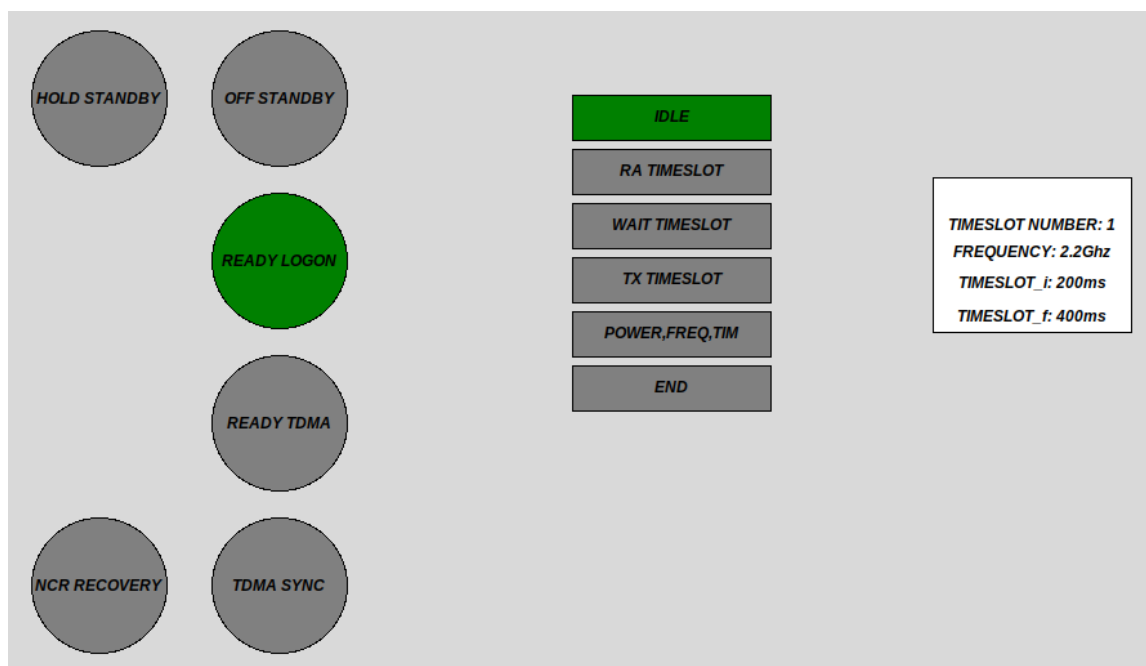


Figura 96 – MF-TDMA: Ready for logon - Timeslot 1

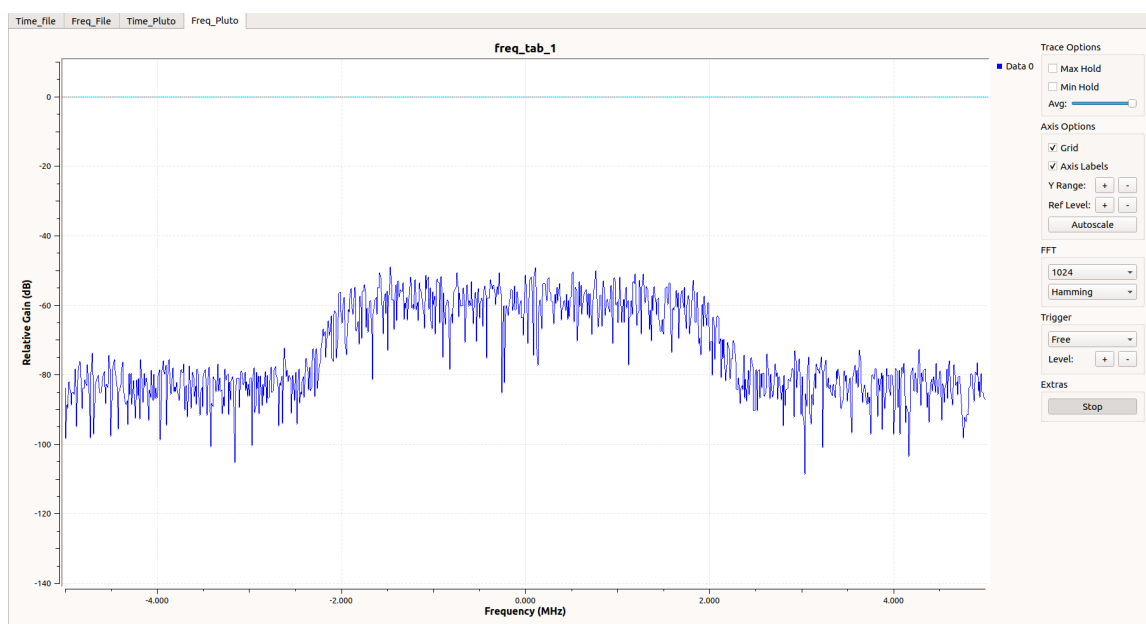


Figura 97 – GNU Radio: Frequencia de RX em 2.2 Ghz - Timeslot 1

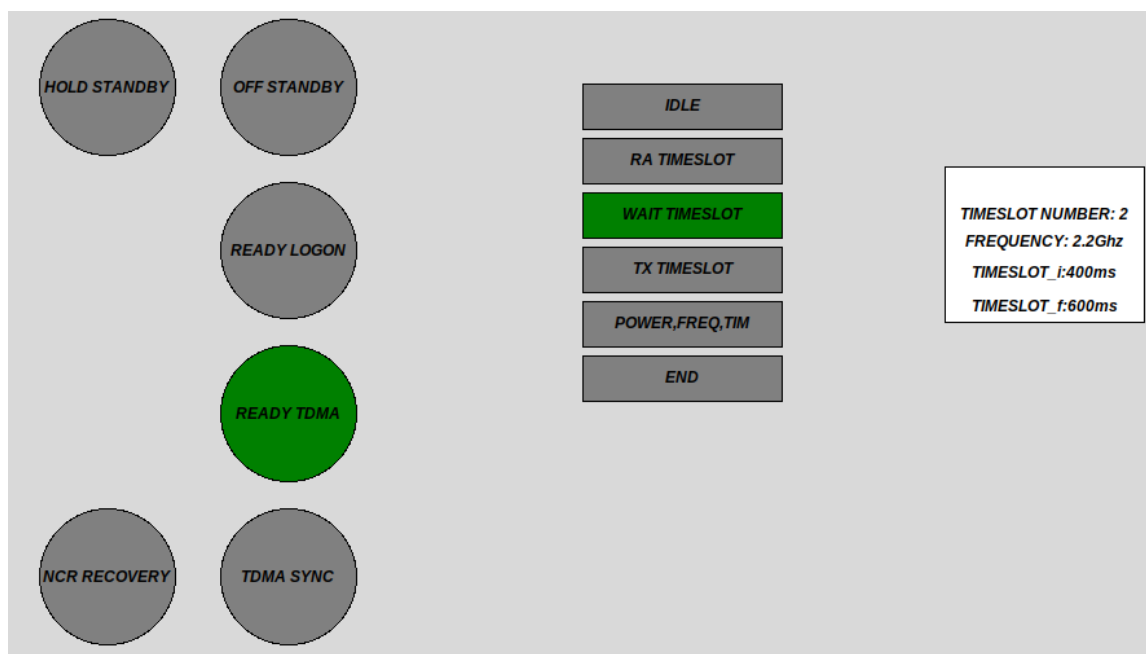


Figura 98 – MF-TDMA: Ready for TDMA - Timeslot 2

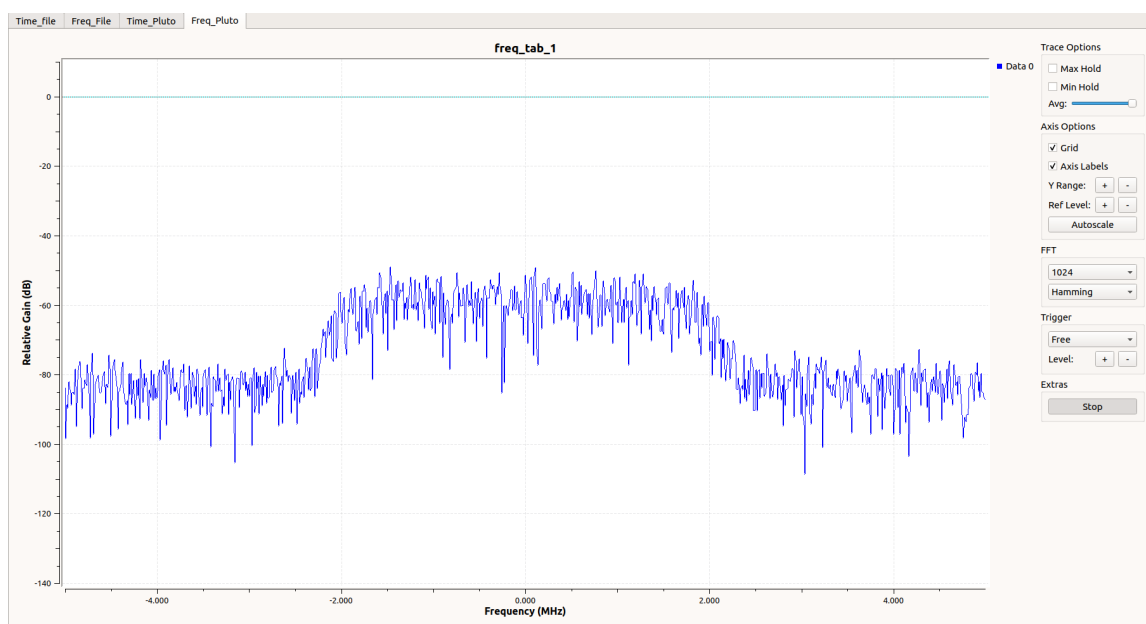


Figura 99 – GNU Radio: Frequencia de RX em 2.2 Ghz - Timeslot 2

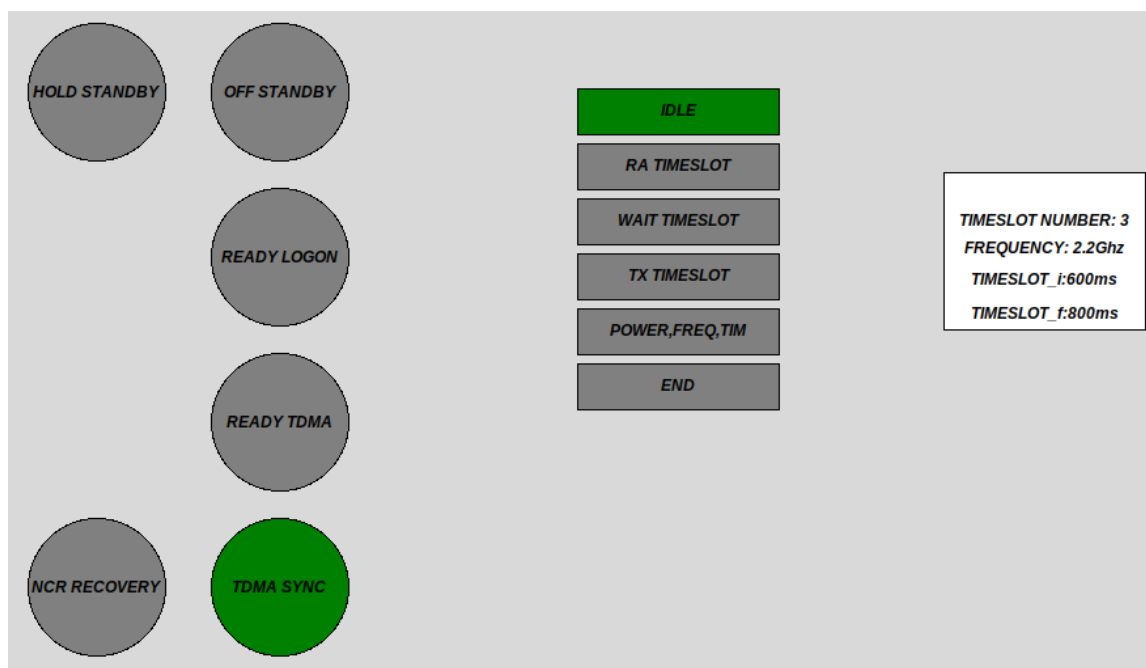


Figura 100 – MF-TDMA: TDMA SYNC - Timeslot 3

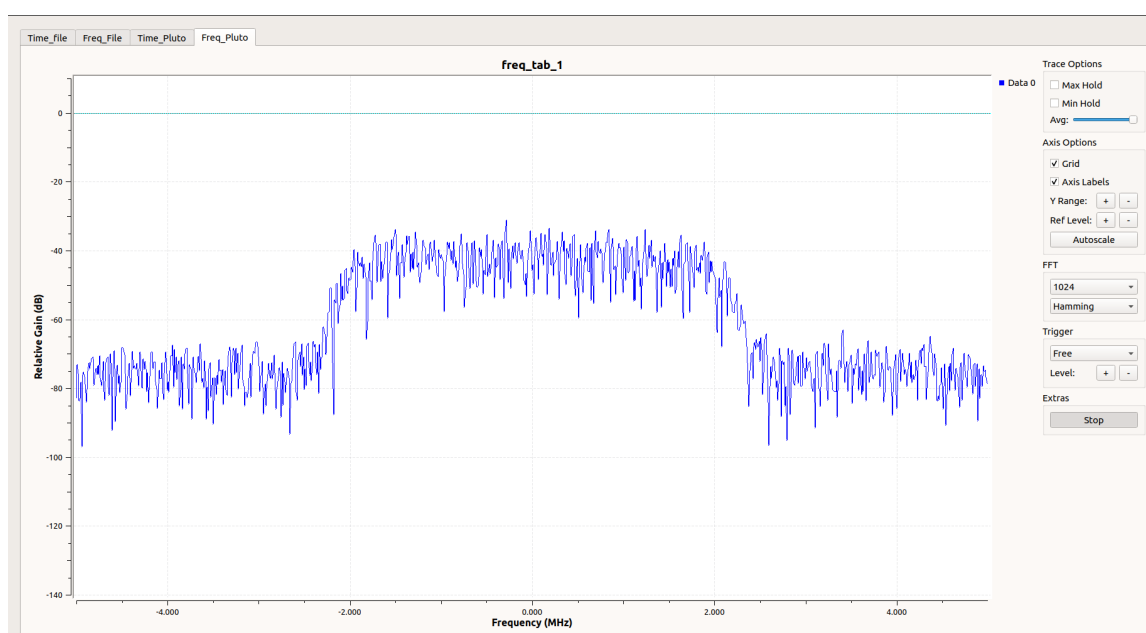


Figura 101 – GNU Radio: Frequencia de RX em 2.2 Ghz - Timeslot 3

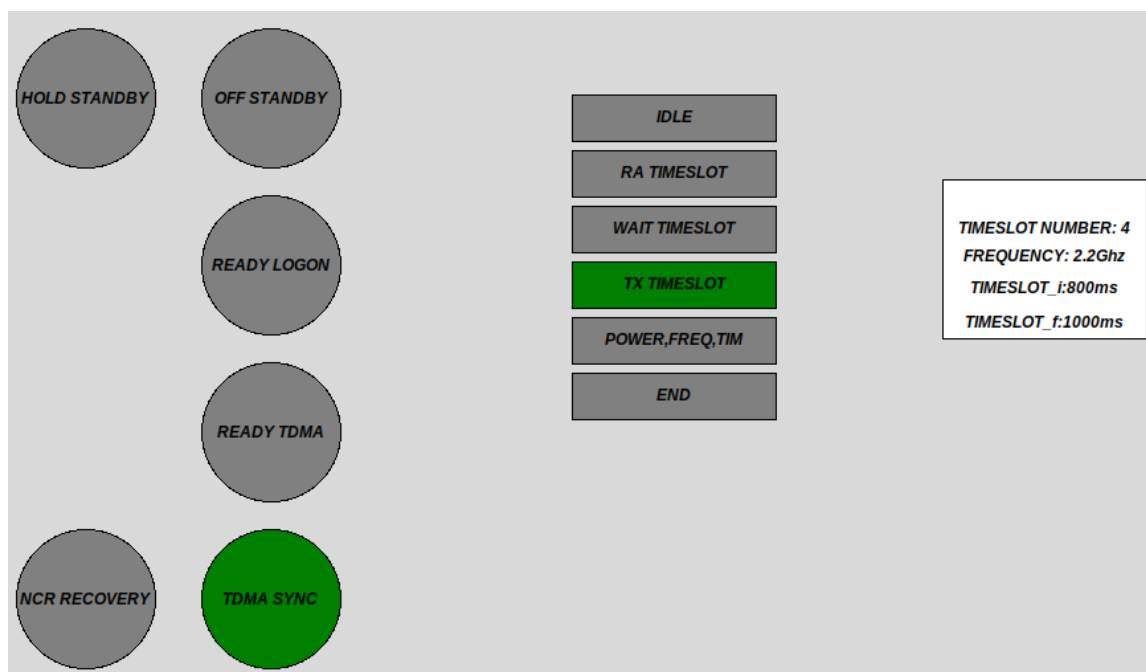


Figura 102 – MF-TDMA:TDMA SYNC - Timeslot 4

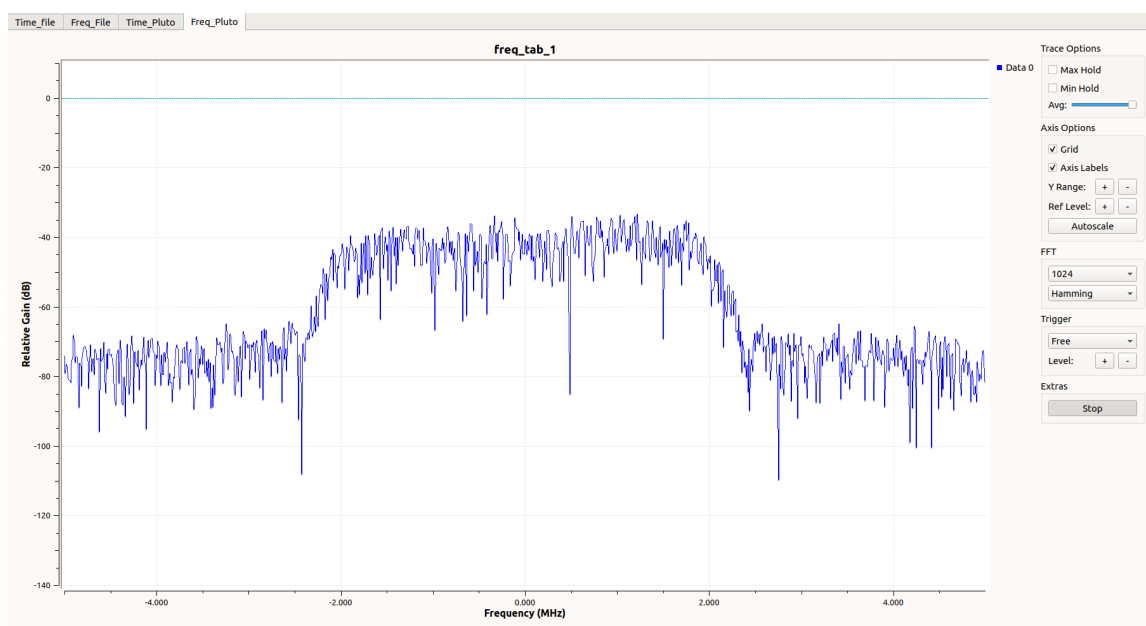


Figura 103 – GNU Radio: Frequencia de RX em 2.2 Ghz - Timeslot 4

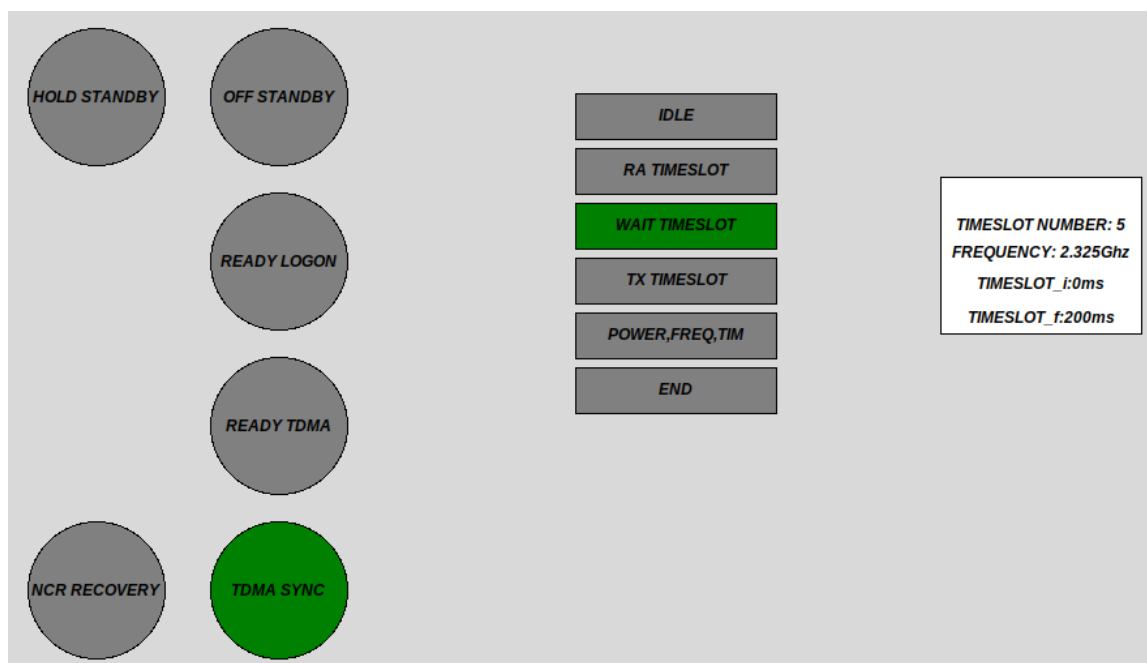


Figura 104 – MF-TDMA: Ready for TDMA - Timeslot 5

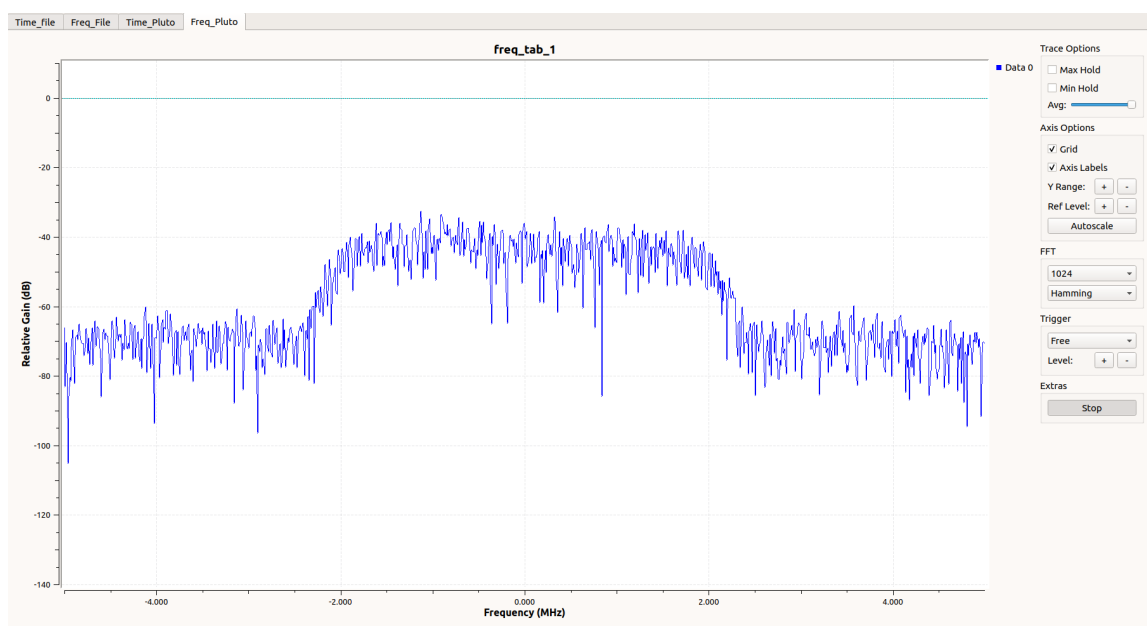


Figura 105 – GNU Radio: Frequencia de RX em 2.2 Ghz - Timeslot 5

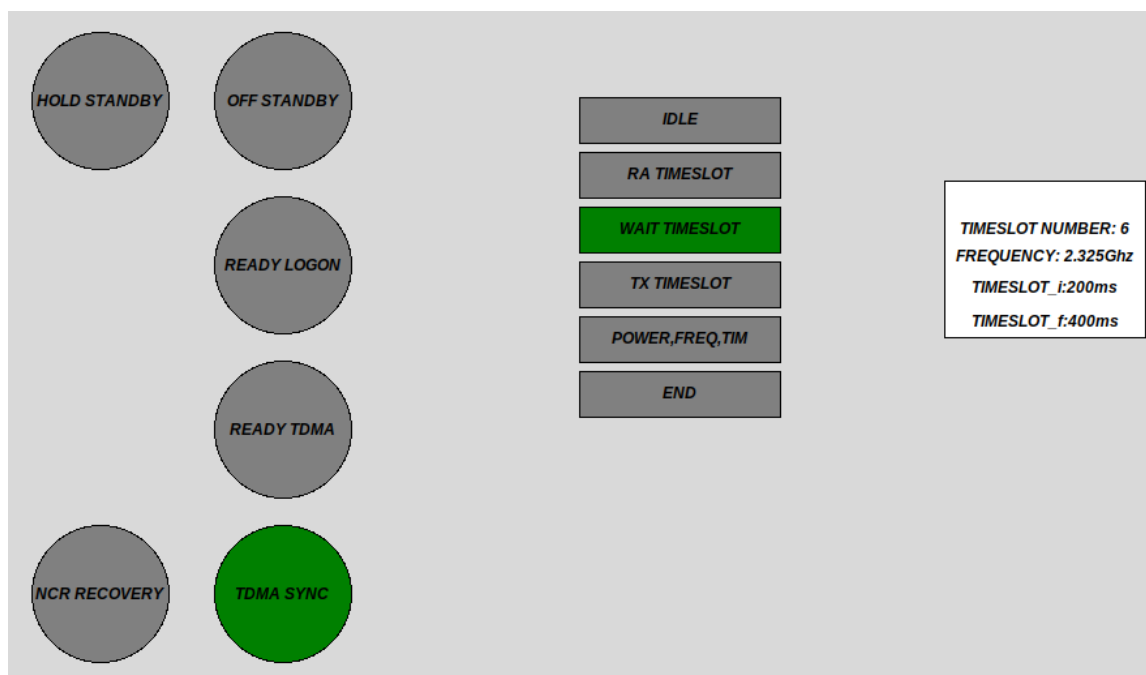


Figura 106 – MF-TDMA: TDMA SYNC - Timeslot 6

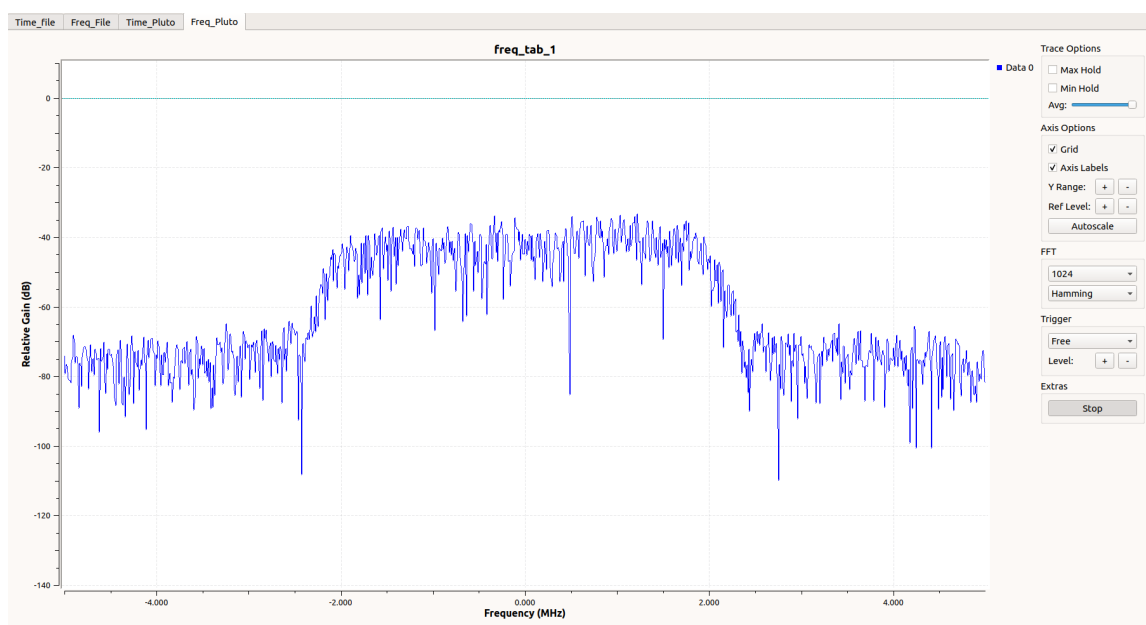


Figura 107 – GNU Radio: Frequencia de RX em 2.325 Ghz - Timeslot 6

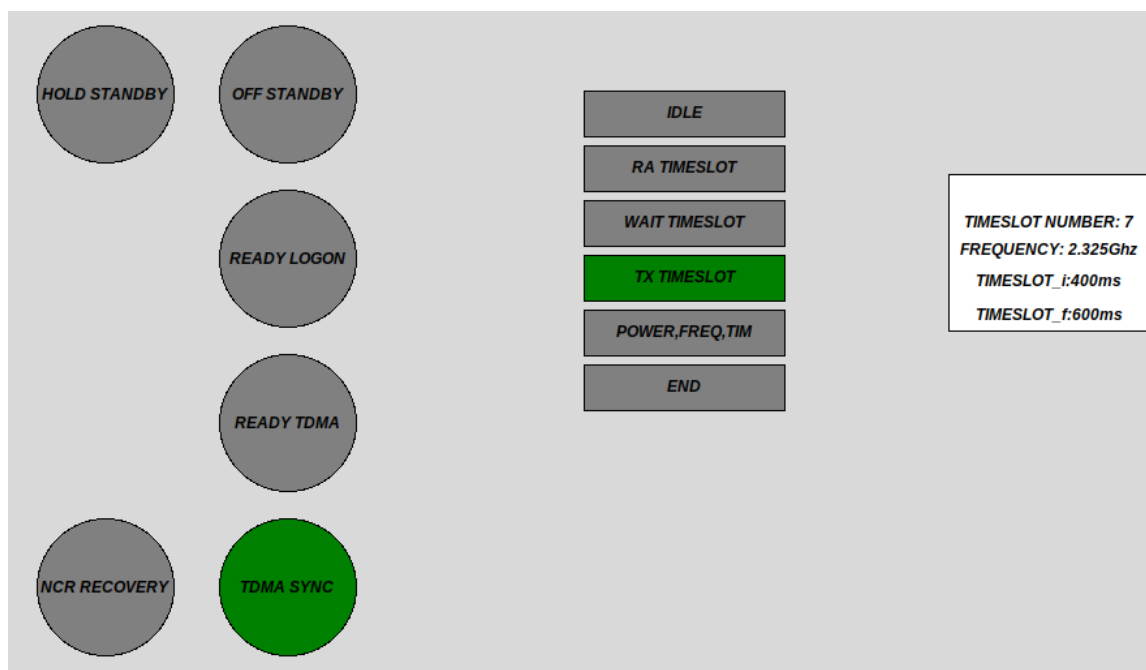


Figura 108 – MF-TDMA: TDMA SYNC - Timeslot 7

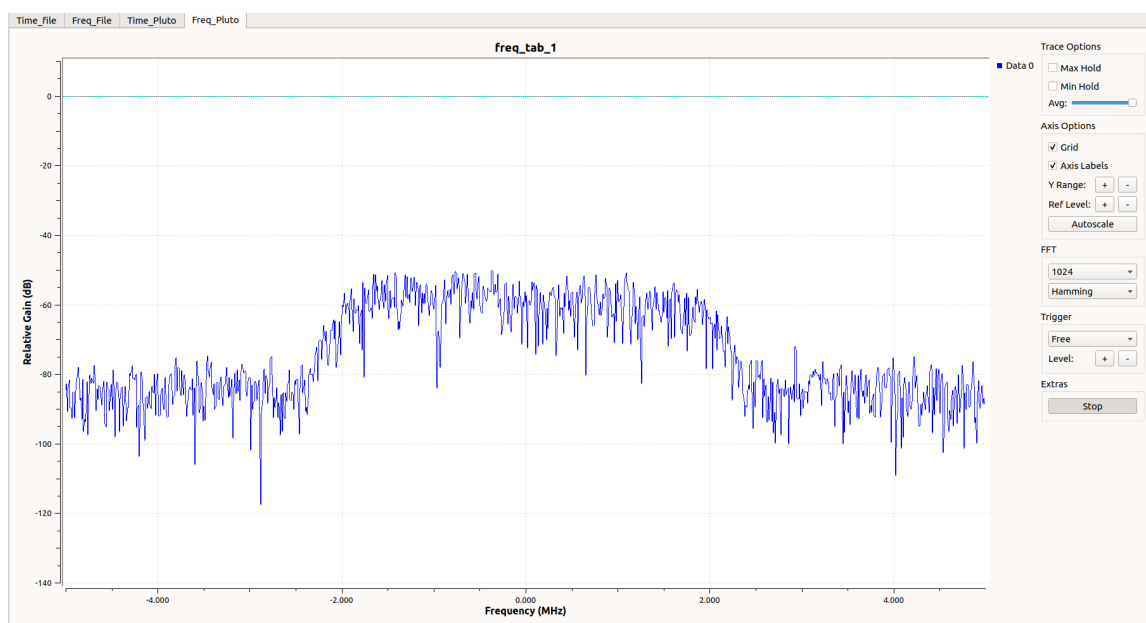


Figura 109 – GNU Radio: Frequencia de RX em 2.325 Ghz - Timeslot 7

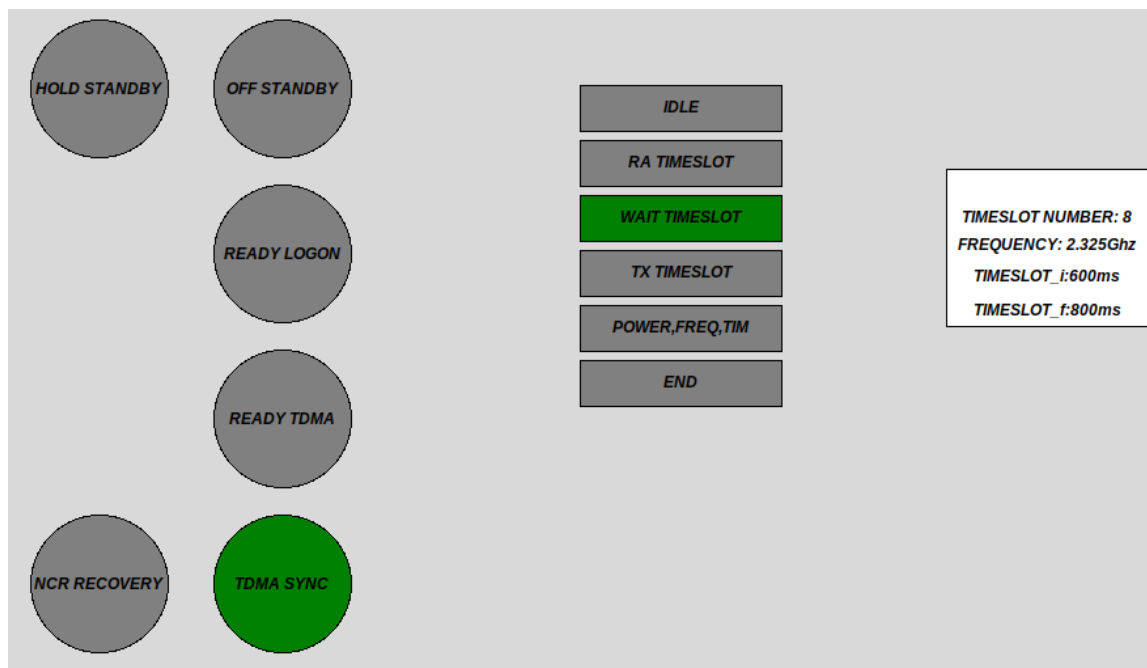


Figura 110 – MF-TDMA: TDMA SYNC - Timeslot 8

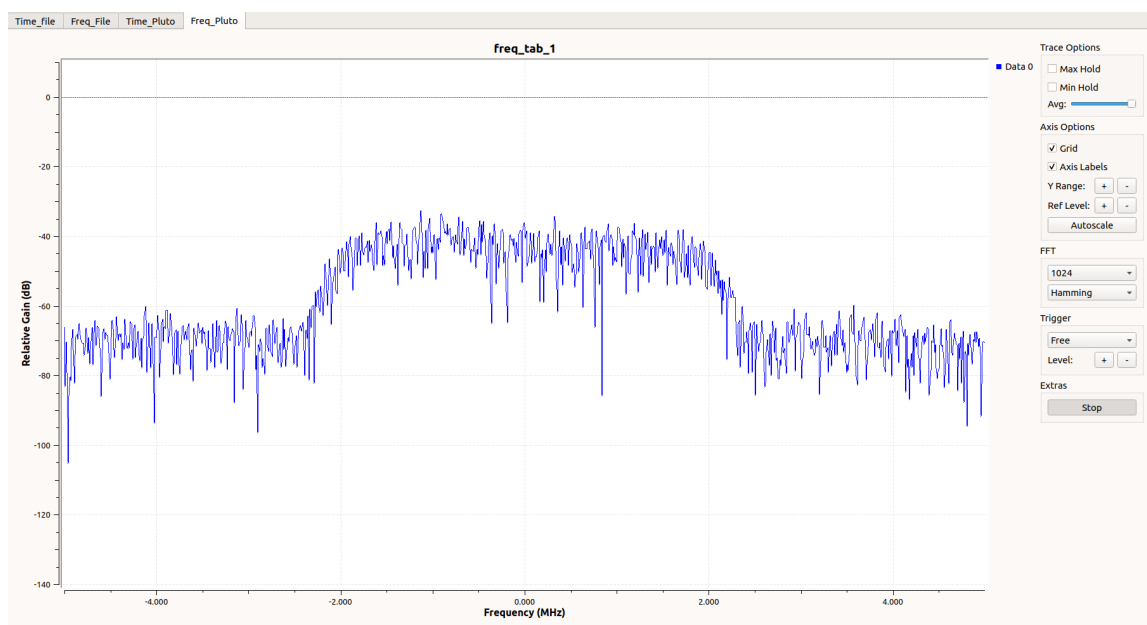


Figura 111 – GNU Radio: Frequencia de RX em 2.325 Ghz - Timeslot 8

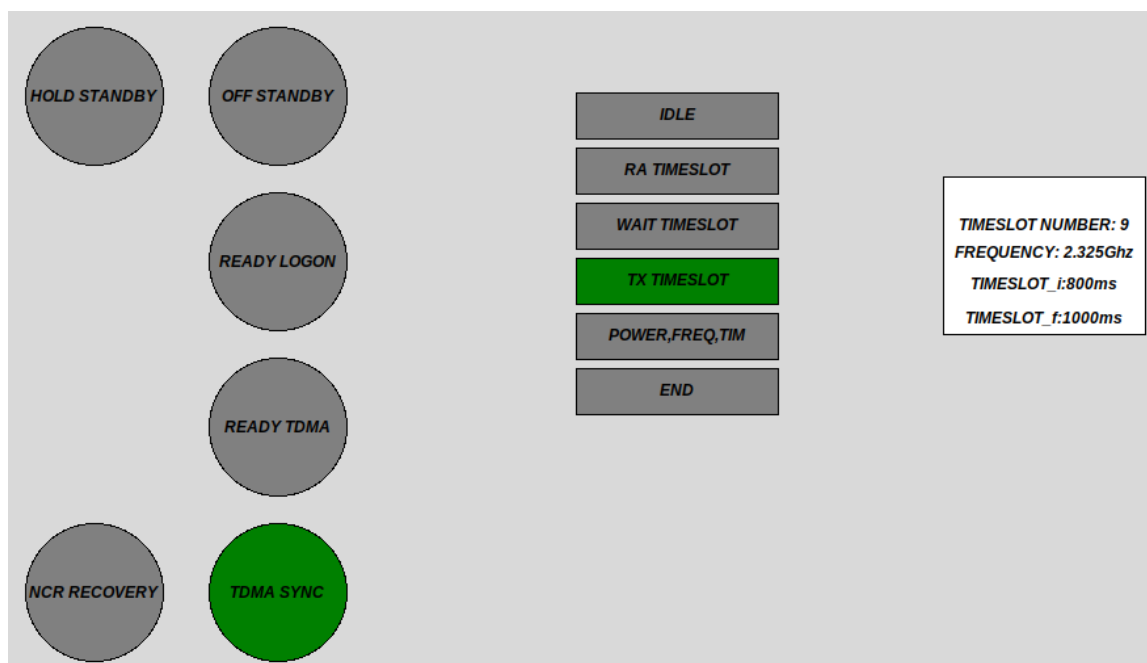


Figura 112 – MF-TDMA: TDMA SYNC - Timeslot 9

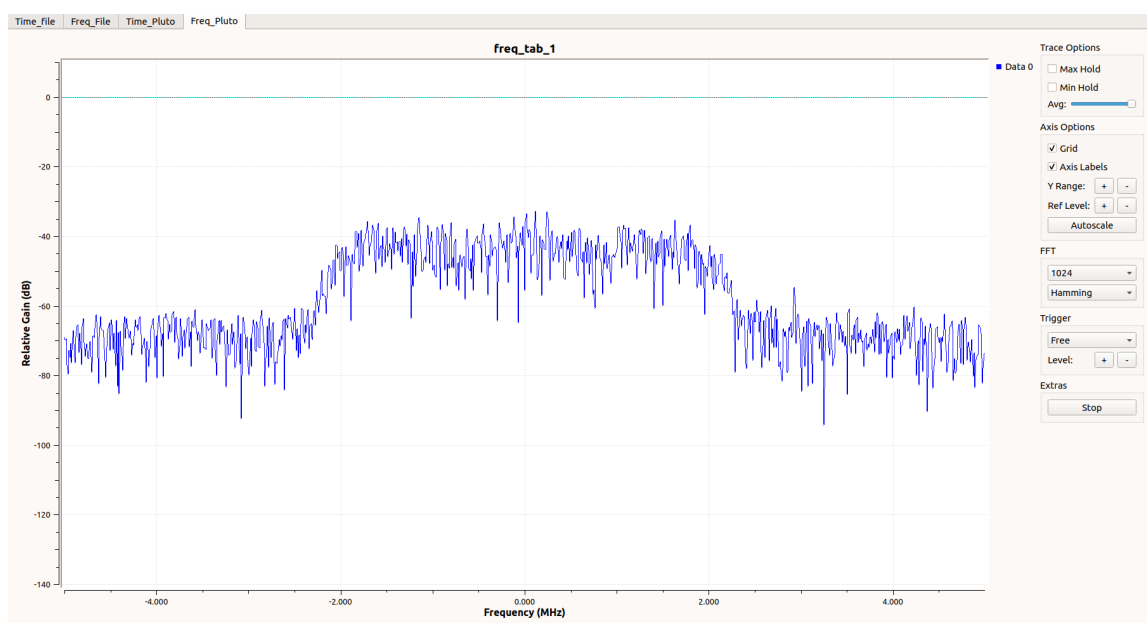


Figura 113 – GNU Radio: Frequencia de RX em 2.325 Ghz - Timeslot 9

Foi possível verificar que os algoritmos do MF-TDMA (vide figuras 38 a 41) foram validados a partir dos blocos presentes na lateral direita de todos os resultados a seguir. Como todos os algoritmos possuem funções compartilhadas, é sempre indicado em verde qual bloco está em execução.

A figura 94 apresenta o estado inicial do MF-TDMA. Após o início do sistema, foi enviada a informação para manter o sinal `TxDisable=1`, utilizando a HUB para enviar uma senoide com frequência de 80KHz. Após modificado o valor de 80KHz para 60KHz, o sistema mudou para o estado de OFF-STANDBY. Ademais, a alteração da condição `TxDisable=1` foi testada para os demais estados, onde foi verificado que o sistema responde da maneira esperada retornando ao estado de HOLD-STADNBY.

A figura 95 apresentam o estado que realiza as preparações para o procedimento de logon. Neste estado foi enviada a informação a partir da HUB para que o sinal `InitLogon` receba-se o valor de 0 ou 1. Desta forma foi possível verificar que o sistema apresentou o comportamento esperado para a transição de estado.

As figuras 96 e 97 apresenta o estado em que foi realizada uma transição do estado *OFF STANDBY* para o estado *READY LOGON*. O sistema recebeu da HUB emulada a confirmação de que a preparação para o logon foi efetuada com sucesso e transitou para o estado *READY LOGON*. Neste estado o sistema realiza uma tentativa de logon no *timeslot* 1, o qual é utilizado para logon. A figura 97 apresenta a janela do GNU Radio recebendo na frequência do *timeslot*.

As figuras 98 e 99 apresentam o estado em que foi realizada uma transição do estado *OFF STANDBY* para o estado *READY LOGON*. O sistema recebeu da HUB a confirmação que a primeira etapa de sincronismo foi efetuada com sucesso e transitou para o estado *READY FOR TDMA SYNC*. Neste estado o sistema recebeu da HUB a solicitação de utilizar o *timeslot* 2, a partir das configurações enviadas na tabela 7. A figura 99 apresenta a janela do GNU Radio recebendo na frequência do *timeslot* 2.

As figuras de 100 a 112 estão sempre no mesmo estado. Neste estado é verificado a reconfiguração dos *timeslots* da HUB a partir da tabela 7. Desta forma, todos os *timeslots* são testados, de forma a apresentar a janela do GNU Radio, com a correta configuração de frequência. Com isto é possível verificar que o sistema está transmitindo na frequência correta o sinal DVB-RCS2.

A validação da transmissão dos *timeslot* no tempo foi feita de maneira separada usando a ESP32. As figuras 114 e 115 apresentam duas abordagens para visualizar a correta transmissão dos *timeslots* no tempo. O led no canto inferior direito da figura 114 irá acender em azul quando for a janela de transmissão do sistema, e será escrito na porta serial a informação *transmitting*, indicando que o sistema está na janela de transmissão do MF-TDMA. O experimento possibilitou verificar que ao observar o led ou a porta serial

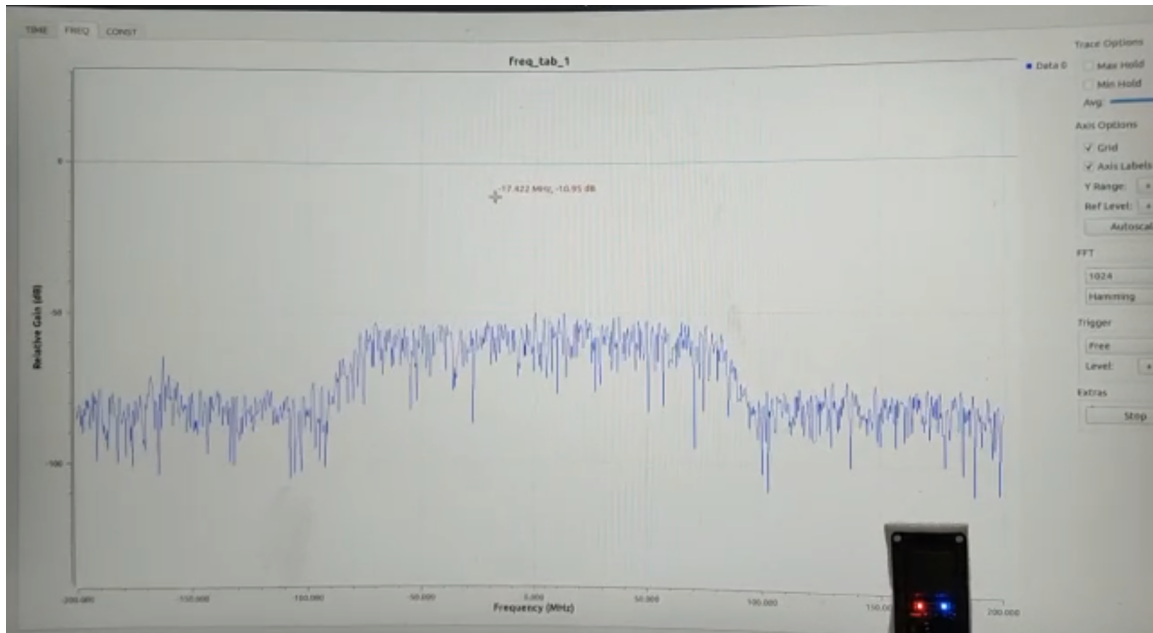


Figura 114 – MF-TDMA: Validação do tempo pelo LED

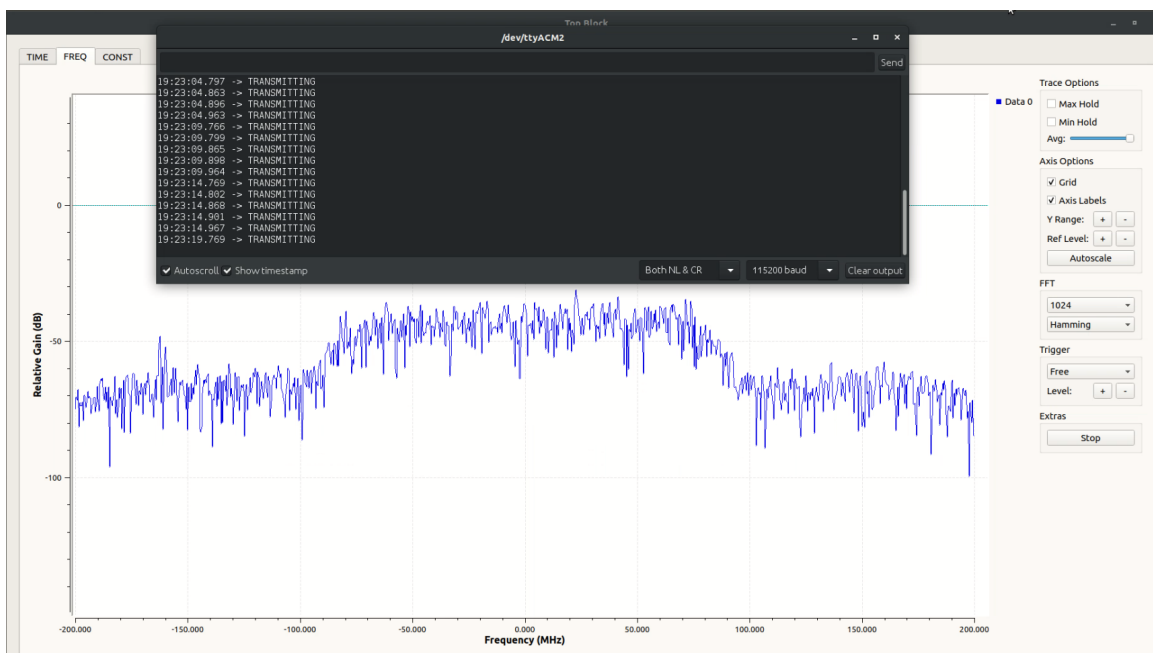


Figura 115 – MF-TDMA: Validação do tempo pelo LED

da ESP32 em conjunto com a tela do GNU Radio, a recepção do GNU Radio, ocorre na janela de tempo indicada por esses 2 métodos, o que indica que o MF-TDMA está transmitindo no tempo correto. O video do experimento pode ser acessado em ([ALVES, 2022](#)).

6 Conclusões

O codificador RCS2 apresentou bons resultados quanto ao tempo de codificação tendo em vista uma implementação em ARM do sistema. Dados os valores em teste, a partir da análise do tempo de codificação foi possível validar o limite máximo de codificação de $40ms$ para o sistema que utiliza o sistema de modulação TC-LM. Quanto a codificação utilizando SS-LM, o tempo de execução obtido foi bem superior, e isto está condizente com o sistema, tendo em vista que este tipo de codificação visa garantir uma melhor proteção dos dados transmitidos. Este sistema é utilizado em casos em que há muita interferência durante a transmissão do *burst*. Desta forma, é possível concluir que o sistema apresentou um resultado condizente com o seu propósito.

A implementação do MF-TDMA ainda que parcial apresentou avanços quanto ao sistema de controle principal do sistema. Foi possível verificar as corretas transições dos estados do sistema, bem como validar a correta leitura e escrita das tabelas e descritores do sistema. O codificador RCS2 apresentou o seu correto funcionamento onde foi possível validar a partir dos modelos implementados o seu correto funcionamento. A validação foi possível através de simulações de constelações, diagramas de olho e tempo de execução.

Quanto às tabelas e descritores do MF-TDMA, foram testadas todas que são relativas ao controle da máquina de estados, e as que utilizam algumas funções internas do sistema. Quanto à FSM a tabela TIM-U, a qual recebe informações sobre as transições de estado do sistema, apresentou o correto funcionamento para todos os seus possíveis valores. As tabelas de construção do *super frame* foram validadas, de forma que a partir de valores inseridos manualmente em cada uma delas, foi possível a correta obtenção das informações presentes nas mesmas.

A validação das funções internas do MF-TDMA deu-se a partir da verificação do correto uso e resposta das mesmas. Quanto aos estados de transmissão, as funções de obtenção dos *timeslots* de controle e acesso aleatório foram completamente validadas a partir dos valores de teste utilizadas nas tabelas e descritores. A partir da ferramenta LabView e python foi possível a verificação do funcionamento do sistema de forma gráfica. A figura 90 apresenta o funcionamento da função de obtenção de um *timeslot* para acesso aleatório, o qual pode ser visto em verde no gráfico representativo do *frame*. Quanto aos *timeslots* de controle, pode ser observado na figura 91 a obtenção do mesmo. As figuras permitem verificar o tipo do *timeslot* utilizado a partir de uma verificação presente, a qual indica em amarelo uma atividade válida.

A validação da transmissão no tempo e na frequência do MF-TDMA foi possível a partir do uso do GNU Radio, onde foi possível verificar a existência de um sinal na

frequência utilizada pelo *timeslots* em uso. A validação no tempo a partir do uso da ESP32 permitiu verificar que a partir de uma referência de tempo externa ao MF-TDMA, a transmissão sempre ocorre no tempo previsto para o timeslot, em ambos os sistemas.

A análise do sincronizador permitiu concluir que o sistema PLL para sincronizar 2 contadores separados funcionou da maneira esperada, o que pode ser constatado a partir da simulação do módulo utilizando a ferramenta do Vivado com o experimento realizada em ARM;

Referências

- ABUOWAIMER DANI MAAROUF, T. M. J. F. G. G. S. A. Z.; VANNELLI, A. *GPlace3.0: Routability-Driven Analytic Placer for UltraScaleFPGA Architectures*. Canada: [s.n.], 2018. Disponível em: <<https://dl.acm.org/doi/10.1145/3233244>>. Citado 3 vezes nas páginas 9, 37 e 38.
- ALVES, B. *ESTADOS DO MFTDMA*. Disponível em: <https://youtu.be/nDDVJ_R4DOY>. Citado na página 90.
- ALVES, B. *Reconfiguração dos timeslots*. Disponível em: <<https://youtu.be/pXKa7ZJFBns>>. Citado na página 90.
- ALVES, B. *Demonstração da transmissão no tempo certo*. 2022. Disponível em: <<https://youtu.be/g8giV2hnsPI>>. Citado na página 103.
- CS457/CS546: Computer Networks II WAN Technologies and Techniques. 2001. Disponível em: <https://www.oocities.org/terry_chim/school/457a1/intro.html>. Citado 3 vezes nas páginas 9, 32 e 33.
- DIGILENT. Disponível em: <<https://reference.digilentinc.com/reference/programmable-logic/zybo/start>>. Citado 2 vezes nas páginas 10 e 60.
- ERL, S.; COLA, T. de. *DVB-RCS2/S2 Testbed: A Distributed Testbed for Next-Generation Satellite System Design and Validation*. Oberpfaffenhofen, Germany, 2014. Disponível em: <<https://ieeexplore-ieee-org.ez54.periodicos.capes.gov.br/stamp/stamp.jsp?tp=&arnumber=6934571>>. Citado 2 vezes nas páginas 9 e 25.
- ETSI. *Digital Video Broadcasting (DVB); Second Generation DVB Interactive Satellite System (DVB-RCS2); Part 1: Overview and System Level specification*. Siret N 348 623 562 00017 - NAF 742 C Association à but non lucratif enregistrée à la Sous-Préfecture de Grasse (06) N 7803/88, 2014–04. Citado 24 vezes nas páginas 7, 8, 9, 10, 21, 22, 24, 28, 29, 30, 31, 33, 34, 35, 36, 39, 40, 47, 48, 49, 51, 54, 56 e 57.
- JEON, D.-G. O. H. Ncr clock recovery method suitable for dvb-s2/rcs2 systems. In: *International Conference on Information and Communication Technology Convergence (ICTC)*. [S.l.: s.n.], 2014. Citado 4 vezes nas páginas 10, 44, 45 e 46.
- JEONG JOONGYU RYU, M. S. H. L. H. Design and implementation of multi-frequency tdma modulator for satellite terminal based on dvb-rcs. In: . ETRI, Korea: [s.n.], 2003. Citado 4 vezes nas páginas 9, 40, 41 e 46.
- JOOST, M. *Theory of Root-Raised Cosine Filter*. 2010. Disponível em: <<https://michael-joost.de/rrfilter.pdf>>. Citado 2 vezes nas páginas 9 e 27.
- LATA, K.; KUMAR, M. *ALL Digital Phase-Locked Loop (ADPLL): A Survey*. 2013. Disponível em: <<http://www.ijfcc.org/papers/225-E353.pdf>>. Citado 2 vezes nas páginas 36 e 37.
- LISS, J. *Implementation of a VBR MPEG-stream receiver in an FPGA*. [S.l.], 2012. Citado 7 vezes nas páginas 9, 10, 44, 45, 46, 56 e 57.

- NETTO, F. abio S. *Geradores de Sequências Pseudoaleatórias Usando Caos em Sistemas de Espalhamento Espectral*. São Paulo: [s.n.], 2009. Citado 2 vezes nas páginas 9 e 32.
- R.PRABHU R.NAGARAJAN, N. S. *Implementation of Direct Sequence Spread Spectrum Communication System Using FPGA*. Namakkal, India, 2017. Citado 4 vezes nas páginas 9, 41, 42 e 46.
- SANTOS, D. A. da S. *Implementação de um Modulador para DVB-RCS2 em FPGA*. Brasília, DF: [s.n.], 2019. Citado na página 31.
- SAROJINI, C. R. *Design and Implementation of DSSS-CDMA Transmitter and Receiver for Reconfigurable Links Using FPGA*. [S.l.], 2012. Citado 4 vezes nas páginas 9, 42, 43 e 46.
- SATEESHKUMAR, H. C. *SRRC Filter Implementation As Per DVB-S2 Standard*. Bangalore, India, 2013. Disponível em: <<http://www.ijfcc.org/papers/225-E353.pdf>>. Citado 4 vezes nas páginas 9, 39, 40 e 46.
- WEI, S. *Under-ice underwater acoustic communication based on direct sequence spread spectrum system with parametric emission*. Shanghai, China: [s.n.], 2016. Disponível em: <<https://ieeexplore-ieee-org.ez54.periodicos.capes.gov.br/stamp/stamp.jsp?tp=&arnumber=7753707>>. Citado na página 32.
- XILINX. Disponível em: <<https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html>>. Citado 2 vezes nas páginas 9 e 38.
- YAZDANI, N. *MULTI-FREQUENCY TIME-DIVISION MULTIPLE-ACCESS (MF-TDMA) RESOURCE PACKING*. Lexington, MA, 2008. Citado na página 34.